## MULTIPLE ZEROS OF POLYNOMIALS

(NASA-CR-139675) MULTIPLE ZEROS OF POLYNOMIALS (Oklahoma State Univ. Research Foundation) 329 p HC \$19.50 CSCL 12A N74-32042

Unclas G3/19 48143

bу

CRAIG A. WOOD

Department of Mathematics Oklahoma State University Stillwater, Oklahoma



A National Aeronautics and Space Administration

Research Grant

Grant Number NASA NGR 37-002-084

[FINAL]

# MULTIPLE ZEROS OF POLYNOMIALS

by-

CRAIG A. WOOD

Department of Mathematics Oklahoma State University Stillwater, Oklahoma

A National Aeronautics and Space Administration

Research Grant

Grant Number NASA NGR 37-002-084

Abstract: MULTIPLE ZEROS OF POLYNOMIALS

Grant: The National Aeronautics and Space Administration

Grant Number: NASA NGR 37-002-084

Grantee: Professor Craig A. Wood

Department of Mathematics Oklahoma State University

Stillwater, Oklahoma

Various classical methods exist for extracting the zeros of a polynomial

$$P(X) = a_1 X^N + a_2 X^{N-1} + ... + a_{N+1}$$

where  $a_1 \neq 0$  and  $a_1, a_2, \dots, a_{N+1}$  are complex numbers, when N=1,2,3,4. For polynomials of higher degree, iterative numerical methods must be used. In this material four iterative methods are presented for approximating the zeros of a polynomial using a digital computer. Newton's method and Muller's method are two well known iterative methods which are presented. They extract the zeros of a polynomial by generating a sequence of approximations converging to each zero. However, both of these methods are very unstable when used on a polynomial which has multiple zeros. That is, either they fail to converge to some or all of the zeros, or they converge to very bad approximations of the polynomial's zeros.

This material introduces two new methods, the greates common divisor (G.C.D.) method and the repeated greatest common divisor (repeated G.C.D.) method, which are superior methods for numerically approximating the zeros of a polynomial having multiple zeros.

The above methods were all programmed in FORTRAN IV and comparisons in time and accuracy are given. These programs were executed on the

IBM 360/50 computer as well as the UNIVAC 1108 and the CDC 6600 computer.

This material also contains complete documentations for six

FORTRAN IV programs. Flow charts, program listings, definition of

variables used in the program, and instructions for use of each program

are included.

#### PREFACE

Four iterative methods for approximating the zeros of a polynomial using a digital computer are presented in this material. Chapter I is an introduction. Chapters II and III contain Newton's and Muller's methods, respectively. Chapters IV and V present two new methods which depend upon finding the greatest common divisor of two polynomials. Chapter VI contains a comparison of the four methods. Flow charts, FORTRAN IV programs, and complete program documentations for these four methods are presented in appendicies A through H.

I would like to express my appreciation to the National Aeronautics and Space Administration, specifically the Manned Spacecraft Center in Houston, Texas, for their financial support in making this work possible under grant number NASA NGR 37-002-084. I would also like to thank Randy Snider, a graduate assistant supported by this grant, for the great deal of work he put in on the FORTRAN programs. In particular, the material on Newton's and Muller's Methods included in this paper is part of his masters thesis at Oklahoma State University.

# TABLE OF CONTENTS

Chapte	er e e e e e e e e e e e e e e e e e e	P	age
I.	INTRODUCTION	D	1
II.	NEWTON'S METHOD	•	4
	Derivation of the Algorithm	0 0 P	8 9 10
III.	MULLER'S METHOD	ē	12
	Derivation of the Algorithm	٥	19 20
IV.	GREATEST COMMON DIVISOR METHOD		22
	Derivation of the Algorithm		29
v.	REPEATED GREATEST COMMON DIVISOR	·	30
	Derivation of the Algorithm	e D	30 32
VI.	CONCLUSION	Ď	34
	Polynomials With all Distinct Roots	8	35 40
REFEREN	NCES	٠	85
APPEND]	IX A - SPECIAL FEATURES OF NEWTON'S AND MULLER'S PROGRAMS	a	86
	Generating Approximations		88

	Chapter	Page
•		Improving Zeros Found 91
		Solving Quadratic Polynomial 91
		Missing Roots
		Miscellaneous
	APPENDIX B -	NEWTON'S METHOD
		Use of the Program
		Input Data for Newton's Method
		Variables Used in Newton's Method 107
		Description of Program Output
		Informative and Error Message 109
		Flow Charts
		Program Listing
		Flogiam risting
	APPENDIX C -	MULLER'S METHOD
		Use of the Program
		Input Data for Muller's Method
		Variables Used in Muller's Method 137
		Description of Program Output
		Informative and Error Messages
		Flow Charte
		Flow Charts
		Flogiam Listing
	APPENDIX D -	SPECIAL FEATURES OF THE G.C.D. AND THE
		REPEATED G.C.D. PROGRAMS
		Generating Approximations
		Altering Approximations
		Conversion the Complex Diseases and a complex Diseases
		Searching the Complex Plane
		Improving Zeros Found
		Solving Quadratic Polynomial
		Missing Roots
		Miscellaneous
	APPENDIX E -	G.C.D NEWTON'S METHOD
		Use of the Program
		Input'Data for G.C.D Newton's Method 173
		Variables Used in G.C.D Newton's Method 178
		Description of Program Output
		Informative and Error Messages
		Flow Charts
		Program Listing

		4	* .*		·	•	
		e de la companya de l	e.				
C)	napter					Page	
A	PPENDIX F	- G.C.D MULL	ER'S METHOD		· · · ·	. 214	
		Use of the Prop	gram			214	
		Input Data for	G.C.D Mulle	er's Method.		. 216	
		Variables Used	in G.C.D Mu	ller's Metho	od	. 216	
	•	Description of	Program Output			. 216	
		Informative and	d Error Message	s		. 216	
		Flow Charts				. 222	
		Program Listing	g			234	
	•	, i	9			. 25.,	
Al	PPENDIX G	- REPEATED G.C.I	D NEWTON'S N	ETHOD		. 253	
		Use of the Prog	gram			. 253	
		Input Data for	Repeated G.C.I	Newton's	Method	. 255	
		Variables Used	in Repeated G.	C.D Newto	n's		
		Method				. 255	
		Description of	Program Output			. 255	
		Informative and	d Error Message	8		256	
		Flow Charts				260	
		Program Listing	7			270	
			<b>5</b>	• • • • • •	• • • •	. 270	
AF	PENDIX H	- REPEATED G.C.I	O MULLER'S M	ETHOD		. 286	
		Use of the Prog Input Data for	Repeated G.C.D	Muller's	<b>.</b>		
		Method				. 287	
		Variables used	in Repated G.C	.D Muller	¹s		
		Method			-	. 288	
	•	Description of	Program Output			- 288	
		Informative and	l Error Message	S		289	
		Flow Charts				200	
		Program Listing	,		• • • •	201	

#### CHAPTER I

#### INTRODUCTION

Frequently in scientific work it becomes necessary to find the zeros, real or complex, of the polynomial of degree N

$$P(X) = a_1 X^N + a_2 X^{N-1} + ... + a_N X + a_{N+1}$$

where  $a_1 \neq 0$  and the coefficients  $a_1, a_2, \dots, a_{N+1}$  are complex numbers. Various classical methods calculate the exact roots of polynomials of degree 1,2,3, or 4. For polynomials of higher degree, no such methods exist. Thus, to solve for the zeros of such polynomials, numerical methods of iteration based on successive approximations must be employed. In the following material four such methods are given which are particularly suited for modern high speed computers.

Newton's method is an iterative procedure which generates a sequence of successive approximations of a zero of P(X) by using the iteration formula

$$X_{n+1} = X_n - P(X_n)/P'(X_n).$$

An initial approximation to the zero is required to start the iterative process. Under certain conditions this sequence will converge quadratically to the desired root. It is, however, necessary to compute the value of the polynomial and its derivative for each step in the

iterative procedure. Once a zero of P(X) has been found, it is divided out of P(X), giving a deflated polynomial of lower degree. P(X) is replaced by the deflated polynomial and the iterative process is applied to extract another zero of P(X). This procedure is repeated until all zeros of P(X) have been found. The zeros may then be rechecked and their accuracy possibly improved by using them as initial approximations with Newton's process applied to the full (undeflated) polynomial.

Muller's method is also an iterative procedure generating a sequence  $X_1, X_2, \ldots, X_n, \ldots$  of successive approximations of a root of P(X). This method converges almost quadratically near a zero and does not require the evaluation of the derivative of the polynomial. Muller's method requires three distinct approximations of a root to start the process of iteration. A quadratic equation is constructed through the three given points as an approximation of P(X). The root of the quadratic closest to  $X_n$  is taken as  $X_{n+1}$ , the next approximation to the zero. This process is then repeated on the last three points of the sequence. After a root of P(X) has been found, P(X) is deflated, and replaced in the above procedure by the deflated polynomial. After all zeros of P(X) are found from successive deflations, they are improved as in Newton's method.

The greatest common divisor method reduces the problem of finding all zeros (possibly multiple zeros) of P(X) to one of extracting the zeros of a polynomial  $P_1(X) = P(X)/D(X)$ , all of whose zeros are simple. D(X), the greatest common divisor of P(X) and its derivative, P'(X), is obtained by repeated application of the division algorithm. Once  $P_1(X)$  is obtained, some suitable method such as Newton's or Muller's method

is used to find the zeros of  $P_1(X)$ . By finding all the zeros of  $P_1(X)$ , all the zeros of P(X) are obtained. The multiplicity of each zero may then be determined.

The repeated greatest common divisor method repeatedly uses the greatest common divisor method to extract the zeros of P(X) and their multiplicities at the same time. That is, the repeated greatest common divisor method reduces the problem of finding the zeros of P(X), which possibly has multiple zeros, to one of finding the zeros of a polynomial which has only simple zeros and the zeros of this polynomial are all the zeros of P(X) of a given multiplicity. The repeated greatest common divisor method must also use a supporting method such as Newton's method or Muller's method.

Chapters II-V contain the examinations of these methods. Each examination includes a development of the method together with the conditions necessary for convergence of the method. Chapter VI contains a comparison of the methods giving advantages and disadvantages of each method.

A complete set of documentations is given for six FORTRAN IV programs in Appendices A-H. Flow charts, program listings, definition of variables used in the program, and instructions for use of each program are included.

It should also be noted that the expressions "zero of a polynomial" and "root of a polynomial" and the words "zero" and "root" are used interchangeably in this material.

#### CHAPTER II

# NEWTON'S METHOD

## 1. Derivation of the Algorithm

Newton's method is probably the most popular iterative procedure for finding the zeros of a polynomial. This fact is due to the excellent results obtained, the simplicity of the computational routine, and the fast rate of convergence obtained provided the initial approximation of a zero is close enough. Also, the method can be applied to the extraction of complex as well as real zeros.

Consider the polynomial

$$P(X) = a_1 X^N + a_2 X^{N-1} + ... + a_N X + a_{N+1}$$
 (2-1)

where  $a_1 \neq 0$  and the coefficients  $a_1, a_2, \ldots, a_{N+1}$  are complex. The algorithm for Newton's method can be derived by approximating P(X) by a Taylor series expansion about an approximation,  $X_0$ , of a zero,  $\alpha$ , of P(X). Using only the first two terms of the expansion, the expression

$$P(X) = P(X_0) + P'(X_0)(X - X_0)$$

is obtained. If this equation is solved for P(X) = 0, then

$$0 = P(X_0) + P'(X_0)(X - X_0)$$

results. Rearranging terms produces

$$0 = P(X_0) + P'(X_0) X - P'(X_0) X_0$$

followed by

$$P'(X_0) X_0 - P(X_0) = P'(X_0) X$$

from which division by P'(X0) produces

$$x_0 - P(x_0)/P'(x_0) = x$$

which is the basic formula for Newton's method. Thus, in general, we obtain the  $(n+1)^{th}$  approximation,  $X_{n+1}$ , of a from the  $n^{th}$  approximation,  $X_n$ , by

$$X_{n+1} = X_n - P(X_n)/P'(X_n).$$
 (2-2)

As a result of repeated use of this algorithm, we obtain the sequence

$$x_0, x_1, x_2, \dots, x_n, \dots$$
 (2-3)

of successive approximations of the root,  $\alpha$ . It should be noted that an initial approximation is necessary to start the iterative process for each new zero; that is, a polynomial of degree N may require N initial approximations.

In order to use equation (2-2), it is necessary to compute, for each  $X_n$ , the value of the polynomial,  $P(X_n)$ , and its derivative,  $P'(X_n)$ . The division algorithm states that if P(X) and G(X) are polynomials, then there exists polynomials H(X) and K(X) such that P(X) = H(X) G(X) + K(X) where K(X) = 0 or deg.  $K(X) < \deg$ . G(X). From this expression of P(X), the following remainder theorem is obtained:

Theorem 2.1. If P(X) is a polynomial and c is a complex number, then the remainder obtained from dividing P(X) by (X - c) is P(c).

The proof of Theorem 2.1 is given in [3, P. 102]. Thus, P(X) can be written as P(X) = (X - c) H(X) + R where P(c) = R. P'(X) is then obtained by the following theorem, the proof of which can be found in [3, PP. 105-106].

Theorem 2.2. If P(X) and H(X) are polynomials and c is a complex number such that P(X) = (X - c) H(X) + R where P(c) = R, then the remainder obtained from dividing H(X) by (X - c) is P'(c).

From synthetic division, an algorithm known as Horner's Method is acquired for computing  $P(X_n)$  and  $P'(X_n)$ .

Theorem 2.3. Let P(X) be defined as in equation (2-1) and let d be a complex number. Define a sequence  $b_1, b_2, \ldots, b_{N+1}$  by

$$b_1 = a_1$$
  
 $b_i = a_i + db_{i-1}$  (i = 2,3,...,N+1).

Define another sequence  $c_1, c_2, \dots, c_N$  by

$$c_1 = b_1$$
 $c_j = b_j + dc_{j-1}$  (j = 2,3,...,N).

Then  $P(d) = b_{N+1}$  and  $P'(d) = c_N$ . The elements  $b_1, b_2, \ldots, b_N$  are the coefficients of the polynomial H(X) in Theorem 2.2 when P(X) is divided by (X - d).

These formulas are derived in [3, PP. 106-107]. Thus with equation (2-2) and the iteration formulas of the previous theorem, Newton's method can now be applied to generate the sequence (2-3) which will converge to the root,  $\alpha$ , if the convergence conditions given in Theorem 2.4 are satisfied.

A criterion is needed to determine when to terminate the sequence (2-3); that is, when has a zero been found? For convergence of the sequence, there must exist a term in the sequence beyond which the difference between any two successive terms is arbitrarily small. Therefore, it is desirable to make the quotient  $|X_n/X_{n+1}|$  sufficiently near 1. From equation (2-2)

$$1 = \left| \frac{X_n}{X_{n+1}} - \frac{P(X_n)}{P'(X_n)} \right|$$

$$\geq \left| \frac{X_n}{X_{n+1}} - \frac{P(X_n)}{P'(X_n)} \right|$$

Thus

$$1 + \frac{\left|\frac{P(X_n)}{P'(X_n)}\right|}{\left|\frac{X_{n+1}}{X_{n+1}}\right|} \ge \left|\frac{X_n}{X_{n+1}}\right|$$

where  $P'(X_n)$  and  $X_{n+1} \neq 0$ . Thus, iterations are continued until an  $X_n$  is obtained such that  $|P(X_n)/P'(X_n)|/|X_{n+1}|$  is as small as desired.

After a zero,  $\alpha$ , of P(X) has been found, the term (X -  $\alpha$ ) is synthetically divided out of P(X) by deflation using Theorem 2.3 obtaining

a polynomial,  $P_1(X)$ , of degree N-1. The root finding process is then repeated to extract a zero,  $\alpha_1$ , of  $P_1(X)$ . P(X) can be written as

$$P(X) = (X - \alpha) P_1(X) + R$$

where  $R = P(\alpha)$ . But  $P(\alpha) = 0$ . Therefore, substitution produces

$$P(X) = (X - \alpha) P_1(X).$$

Now  $P_1(\alpha_1) = 0$  implies that  $P(\alpha_1) = 0$ . Hence,  $\alpha_1$  is a zero of P(X).

By the process of root finding and successive deflations, zeros  $\alpha_0, \alpha_1, \dots, \alpha_{N-1}$  of the deflated polynomials

$$P(X) = P_0(X), P_1(X), \dots, P_{N-1}(X),$$

respectively, are extracted. Each  $\alpha_1$  (1 = 0,1,2,...,N-1) is a zero of P(X) since each  $\alpha_1$  is a zero of P<sub>1-1</sub>(X),P<sub>1-2</sub>(X),...,P<sub>1</sub>(X),P(X).

After all zeros of P(X) have been found, it may be possible to improve their accuracy by using them as initial approximations with Newton's method applied to the full (undeflated) polynomial, P(X). This should correct any loss of accuracy which may have resulted from the successive deflations.

# Convergence of Newton's Method

The following theorem from [2, PP. 79-81] gives sufficient conditions for the convergence of sequence (2-3).

Theorem 2.4. Let P(X) be a polynomial and let the following conditions be satisfied on the closed interval [a,b]:

- 1. P(a) P(b) < 0
- 2.  $P'(X) \neq 0$ ,  $X \in [a,b]$
- 3. P''(X) is either  $\geq 0$  or  $\leq 0$  for all  $X \in [a,b]$
- 4. If c denotes the endpoint of [a,b] at which |P'(X)| is smaller, then  $|P(c)/P'(c)| \le b a$ .

Then Newton's method converges to the (only) solution, s, of P(X) = 0 for any choice of  $X_0$  in [a,b].

When convergence is obtained, it is quadratic; that is,

$$e_{i+1} = \frac{1}{2} F''(\eta_i) e_i^2$$

where  $F(X_1) = X_1 - P(X_1)/P'(X_1)$ ,  $n_1$  is between  $X_1$  and the zero,  $\alpha$ , and  $e_1$  is the error in  $X_1$ . This means that the error obtained in the  $(i+1)^{th}$  iteration of Newton's algorithm is proportional to the square of the error obtained in the  $i^{th}$  iteration. A proof of quadratic convergence can be found in [1, PP. 31-33].

## 3. Procedure for Newton's Method

The general procedure for applying Newton's method is enumerated sequentially as follows, starting with initial approximation  $X_0$ :

1. Calculate a new approximation  $X_{n+1}$  by  $X_{n+1} = X_n - P(X_n)/P'(X_n).$ 

2. Test for convergence; that is, test

$$|P(X_n)/P'(X_n)|/|X_{n+1}| < \varepsilon$$

for some  $\varepsilon$  chosen as small as desired.

3. If convergence is obtained, perform the following:

- a. Save  $X_{n+1}$  as the desired approximation to a zero of P(X).
- b. Deflate P(X) using  $X_{n+1}$ .
- c. Replace P(X) by the deflated polynomial.
- d. Return to step 1 with a new initial approximation.
- 4. If no convergence is obtained, increase n by 1 and return to step 1.

In order to prevent an unending iteration process in case the method does not produce convergence, a maximum number of iterations should be specified. If convergence is not obtained within this number of iterations, change the initial approximation and return to step 1 above.

4. Geometrical Interpretation of Newton's Method

A geometrical interpretation of Newton's method is given in Figure 2.1. $X_1$  is an approximation to the zero,  $\alpha$ .  $P'(X_1)$  is the slope of the line tangent to P(X) at  $X_1$ .  $X_{1+1}$  is the intersection of the tangent line with the x axis.

### 5. Determining Multiple Roots

 $\mathbb{H}^{-}\mathbb{P}(X)$  has m distinct zeros, then  $\mathbb{P}(X)$  can be written as

$$P(X) = a_1(X - \alpha_1)^{e_1} (X - \alpha_2)^{e_2} ... (X - \alpha_m)^{e_m}, (m \le N)$$

where  $\alpha_1$  is a zero of P(X) and  $e_i$  is the multiplicity of  $\alpha_1$  (i = 1,2,...,m). Consider the root  $\alpha_j$ . Dividing out the term

 $(X - \alpha_j)$  by deflating P(X) gives  $P_1(X)$  of degree N-1 which can be written as

$$P_1(X) = (X - \alpha_1)^{e_1} (X - \alpha_2)^{e_2} ... (X - \alpha_j)^{e_j^{-1}} ... (X - \alpha_m)^{e_m}$$

Evaluating  $P_1(X)$  at the zero,  $\alpha_j$ , gives  $P_1(\alpha_j) = 0$  if  $e_j > 1$ . Thus, after a zero,  $\alpha$ , of P(X) is determined by Newton's iterative process and the current polynomial is deflated giving  $P_1(X)$ , then  $P_1(\alpha)$  is evaluated. If  $P_1(\alpha) \le \varepsilon$  for some small number  $\varepsilon$ ,  $\alpha$  is a root of  $P_1(X)$  and thus has multiplicity at least equal to two.  $P_1(X)$  is then deflated giving  $P_2(X)$ . If  $P_2(\alpha) \le \varepsilon$ ,  $\alpha$  is of multiplicity at least three. This process is continued until a deflated polynomial  $P_k(X)$  is encountered such that either deg.  $P_k(X) = 0$  or  $P_k(\alpha) > \varepsilon$ .  $\alpha$  is then a zero of multiplicity k+1.

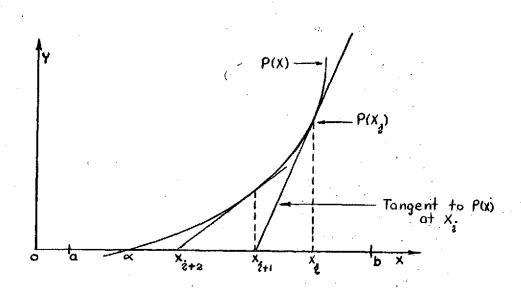


Figure 2.1. Geometrical Interpretation of Newton's Method

#### CHAPTER III

## MULLER'S METHOD

### 1. Derivation of the Algorithm

Muller's method in [4] is an iterative procedure designed to find any prescribed number of zeros, real or complex, of a polynomial. The method does not require the evaluation of the derivative and near a zero the convergence is almost quadratic.

Consider the polynomial

$$P(X) = a_1 X^N + a_2 X^{N-1} + ... + a_N X + a_{N+1}$$
 (3-1)

with complex coefficients such that  $a_1 \neq 0$ . Given three distinct approximations,  $X_{n-2}, X_{n-1}, X_n$ , to a root,  $\alpha$ , of P(X), the problem is to determine  $X_{n+1}$  in such a way as to generate a sequence

$$X_1, X_2, X_3, \dots, X_n, X_{n+1}, \dots$$
 (3-2)

of approximations converging to  $\alpha$ . The points  $(X_{n-2}, P(X_{n-2}))$ ,  $(X_{n-1}, P(X_{n-1}))$ , and  $(X_n, P(X_n))$  determine a unique quadratic polynomial, Q(X), approximating P(X) in the vicinity of  $X_{n-2}, X_{n-1}, X_n$ . A general proof of this can be found in [2, PP. 133-134]. Thus, the zeros of Q(X) will be approximations of the zeros of P(X) in this region of approximation. From the general representation in [2, P. 184] of the Legrangian interpolating polynomial, the representation of Q(X) is given by

$$Q(X) = \frac{(X - X_{n-1})(X - X_{n-2})}{(X_n - X_{n-1})(X_n - X_{n-2})} P(X_n)$$

$$+ \frac{(X - X_n)(X - X_{n-2})}{(X_{n-1} - X_n)(X_{n-1} - X_{n-2})} P(X_{n-1})$$

$$+ \frac{(X - X_n)(X - X_{n-1})}{(X_{n-2} - X_n)(X_{n-2} - X_{n-1})} P(X_{n-2})$$

which can be rewritten as

$$Q(X) = Q(X - X_{n} + X_{n})$$

$$= \frac{(X - X_{n} + X_{n} - X_{n-1})(X - X_{n} + X_{n} - X_{n-1} + X_{n-1} - X_{n-2})}{(X_{n} - X_{n-1})(X_{n} - X_{n-1} + X_{n-1} - X_{n-2})} P(X_{n})$$

$$- \frac{(X - X_{n})(X - X_{n} + X_{n} - X_{n-1} + X_{n-1} - X_{n-2})}{(X_{n} - X_{n-1})(X_{n-1} - X_{n-2})} P(X_{n-1})$$

$$+ \frac{(X - X_{n})(X - X_{n} + X_{n} - X_{n-1})}{(X_{n} - X_{n-1} + X_{n-1} - X_{n-2})(X_{n-1} - X_{n-2})} P(X_{n-2}).$$

In order to simplify this expression, introduce the quantities

$$h_n = X_n - X_{n-1}, h = X - X_n,$$

Then

$$Q(X) = Q(X_n + h)$$

$$= \frac{(h + h_n)(h + h_n + h_{n-1})}{h_n(h_n + h_{n-1})} P(X_n)$$

$$- \frac{h(h + h_n + h_{n-1})}{h_n h_{n-1}} P(X_{n-1})$$

$$+ \frac{h(h + h_n)}{(h_n + h_{n-1})h_{n-1}} P(X_{n-2})$$

$$= \frac{h^2 + 2hh_n + hh_{n-1} + h_n^2 + h_nh_{n-1}}{h_n^2 + h_nh_{n-1}} P(X_n)$$

$$- \frac{h^2 + hh_n + hh_{n-1}}{h_n^2 + h_n^2} P(X_{n-1})$$

$$+ \frac{h^2 + hh_n}{h_n^2 + h_{n-1}^2} P(X_{n-2}).$$

Collecting terms containing like powers of h produces

$$\begin{split} &Q(X) = Q(X_n + h) \\ &= \left( \frac{P(X_n)}{h_n^2 + h_n h_{n-1}} - \frac{P(X_{n-1})}{h_n h_{n-1}} + \frac{P(X_{n-2})}{h_n h_{n-1} + h_{n-1}^2} \right) h^2 \\ &+ \left( \frac{(2h_n + h_{n-1}) P(X_n)}{h_n^2 + h_n h_{n-1}} - \frac{(h_n + h_{n-1}) P(X_{n-1})}{h_n h_{n-1}} + \frac{h_n P(X_{n-2})}{h_n h_{n-1} + h_{n-1}^2} \right) h \\ &+ \frac{h_n (h_n + h_{n-1}) P(X_n)}{h_n^2 + h_n h_{n-1}} \\ &= \left( \frac{P(X_n) h_{n-1}}{h_n h_{n-1} + h_n h_{n-1}} - \frac{P(X_{n-1})}{h_n h_{n-1}} + \frac{P(X_{n-2})}{h_n h_{n-1} + h_{n-1}^2} \right) h^2 \\ &+ \left( \frac{(2h_n h_{n-1} + h_{n-1}^2) P(X_n)}{h_n^2 h_{n-1} + h_n h_{n-1}^2} - \frac{(h_n + h_{n-1}) P(X_{n-1})}{h_n h_{n-1}} + \frac{h_n P(X_{n-2})}{h_n h_{n-1} + h_{n-1}^2} \right) h \end{split}$$

$$+ \frac{(h_n^2 h_{n-1} + h_{n-1}^2 h_n) P(X_n)}{h_n^2 h_{n-1} + h_n^2 h_{n-1}} .$$

Using the common denominator,  $h_n^2 h_{n-1} + h_n h_{n-1}^2$ , and combining terms yields

$$\begin{split} Q(X_n + h) &= \left(\frac{P(X_n) \cdot h_{n-1} - P(X_{n-1}) \cdot (h_n + h_{n-1}) + P(X_{n-2}) \cdot h_n}{h_n^2 h_{n-1} + h_n h_{n-1}^2}\right) h^2 \\ &+ \left(\frac{(2h_n h_{n-1} + h_{n-1}^2) \cdot P(X_n) - (h_n + h_{n-1})^2 \cdot P(X_{n-1}) + h_n^2 P(X_{n-2})}{h_n^2 h_{n-1} + h_n h_{n-1}^2}\right) h^2 \\ &+ \frac{(h_n^2 h_{n-1} + h_{n-1}^2 h_n) \cdot P(X_n)}{h_n^2 h_{n-1} + h_n h_{n-1}^2} \ . \end{split}$$

Multiplying by  $h_n/h_{n-1}^2$  results in

$$Q(X_{n} + h) = \frac{\left[P(X_{n}) \frac{h_{n}}{h_{n-1}} - P(X_{n-1}) \left( \left(\frac{h_{n}}{h_{n-1}}\right)^{2} + \frac{h_{n}}{h_{n-1}} \right) + P(X_{n-2}) \left(\frac{h_{n}}{h_{n-1}}\right)^{2}}{\frac{h_{n}}{h_{n-1}} + h_{n}^{2}} \right] h^{2} + \frac{\left[\left(\frac{h_{n}}{h_{n-1}}\right)^{2} + h_{n}\right] P(X_{n}) - h_{n} \left[\left(\frac{h_{n}}{h_{n-1}}\right) + \left(\frac{h_{n-1}}{h_{n-1}}\right)\right]^{2} P(X_{n-1}) + \frac{h_{n}^{3}}{h_{n-1}^{2}} P(X_{n-2}) h^{2}}{\frac{h_{n}^{3}}{h_{n-1}} + h_{n}^{2}} h^{2}$$

$$+ \frac{\begin{bmatrix} h_{n}^{3} & h_{n-1}^{2} + h_{n}^{2} \\ h_{n-1}^{3} & h_{n}^{3} \\ \frac{h_{n-1}^{3} + h_{n}^{2} \\ \end{bmatrix}}{}.$$

Let 
$$q_n = \frac{h_n}{h_{n-1}}$$
 and  $q = \frac{h}{h_n}$ . Then

$$Q(X_{n} + h) = \left(\frac{P(X_{n}) q_{n} - P(X_{n-1})(q_{n}^{2} + q_{n}) + P(X_{n-2}) q_{n}^{2}}{q_{n}^{+1}}\right) q^{2}$$

$$+ \left(\frac{(2 q_{n}^{+1}) P(X_{n}) - (q_{n}^{+1})^{2} P(X_{n-1}) + q_{n}^{2} P(X_{n-2})}{q_{n}^{+1}}\right) q$$

$$+ \frac{(q_{n}^{+1}) P(X_{n})}{q_{n}^{+1}}.$$

Now let

$$A_{n} = q_{n} P(X_{n}) - q_{n}(q_{n}+1) P(X_{n-1}) + q_{n}^{2} P(X_{n-2})$$

$$B_{n} = (2q_{n}+1) P(X_{n}) - (q_{n}+1)^{2} P(X_{n-1}) + q_{n}^{2} P(X_{n-2})$$

$$C_{n} = (q_{n}+1) P(X_{n}).$$

Then

$$Q(X_n + h) = Q(X_n + qh_n)$$

and

$$Q(X_n + qh_n) = \frac{A_n q^2 + B_n q + C_n}{q_n + 1}$$
.

Solving the quadratic equation  $Q(X_n + qh_n) = 0$  and denoting the result by  $q_{n+1}$  gives:

$$q_{n+1} = \frac{-B_n \pm \sqrt{B_n^2 - 4A_nC_n}}{2A_n}$$

and the new approximation is found as follows:

$$q_{n+1} = \frac{h_{n+1}}{h_n} = \frac{X_{n+1} - X_n}{h_n}$$
.

Thus

$$X_{n+1} = X_n + h_n q_{n+1}$$

In order to avoid loss of accuracy,  $q_{n+1}$  can be written in a better form as follows:

$$q_{n+1} = \frac{\frac{-B_n + \sqrt{B_n^2 - 4A_nC_n}}{2A_n}}{\frac{2A_n}{B_n + \sqrt{B_n^2 - 4A_nC_n}}} \cdot \frac{\frac{B_n + \sqrt{B_n^2 - 4A_nC_n}}{B_n + \sqrt{B_n^2 - 4A_nC_n}}}{\frac{2A_n (B_n + \sqrt{B_n^2 - 4A_nC_n})}{2A_n (B_n + \sqrt{B_n^2 - 4A_nC_n})}}$$

$$q_{n+1} = \frac{\frac{-2C_n}{B_n + \sqrt{B_n^2 - 4A_nC_n}}}{\frac{-2C_n}{B_n + \sqrt{B_n^2 - 4A_nC_n}}} \cdot (3-3)$$

The sign in the denominator should be chosen such that the magnitude of the denominator is largest, thus causing  $|\mathbf{q}_{n+1}|$  to be smallest. This, in turn, will make  $\mathbf{X}_{n+1}$  closest to  $\mathbf{X}_n$ .

Note that each iteration of this process requires three approximations,  $X_{n-2}, X_{n-1}, X_n$ , in order to compute  $X_{n+1}$ . Thus, when  $X_{n+1}$  is found,  $X_{n-1}, X_n, X_{n+1}$  are used to compute  $X_{n+2}$ ; that is, the last three terms of the generated sequence are used to compute the next term.

Convergence of the sequence (3-2) to a zero is obtained when the elements  $\mathbf{X}_k$  and  $\mathbf{X}_{k+1}$  of the sequence are found such that

$$\frac{|X_{k+1} - X_k|}{|X_{k+1}|} < \epsilon, X_{k+1} \neq 0;$$

that is, the ratio of the change in the approximation to the approximation itself is as small as desired.

In order to use the iterative formulas, it is necessary to compute the value,  $P(X_j)$ , of the polynomial P(X) at the approximation  $X_j$ . The procedure for doing this is discussed in Chapter II, § 1. The iteration formulas are given in Theorem 2.3 of Chapter II.

After a zero,  $\alpha$ , of P(X) has been found, P(X) is deflated as described in Chapter II, § 1, and the process repeated to extract a zero,  $\alpha_1$ , of P<sub>1</sub>(X). By applying Muller's method to successively deflated polynomials, all the zeros of P(X) are obtained. For more detailed discussion of this procedure see Chapter II, § 1, keeping in mind that Muller's instead of Newton's method is used.

Muller's method requires three initial approximations to a zero in order to start the iteration process. If three are not known, the values  $X_1 = -1$ ,  $X_2 = 1$ ,  $X_3 = 0$  can be used.

Convergence of Muller's method is almost quadratic provided the three initial approximations are sufficiently close to a zero of P(X). This is natural to expect since P(X) is being approximated by a

quadratic polynomial. Quadratic convergence means that the error obtained in the (n+1)<sup>th</sup> step of the iterative process is proportional to the square of the error obtained in the n<sup>th</sup> iteration. However, no general proof of convergence has been obtained for Muller's method. It has produced convergence in the majority of the cases tested.

In application of Muller's method, an alteration should be made to handle the case in which the denominator of equation (3-3) is zero (0). This occurs whenever  $P(X_n) = P(X_{n-1}) = P(X_{n-2})$ . If this happens, set  $q_{n+1} = 1$ .

Another alteration which should be made in actual practice is to compute the quantity  $|P(X_{n+1})|/|P(X_n)|$  whenever the value  $P(X_{n+1})$  is calculated. If the former quantity exceeds ten (10),  $q_{n+1}$  is halved and  $h_n$ ,  $X_{n+1}$ , and  $P(X_{n+1})$  are recomputed accordingly.

## 2. Procedure for Muller's Method

The basic steps performed by Muller's method are listed sequentially as follows, starting with initial approximations  $X_1$ ,  $X_2$ , and  $X_3$ .

- 1. Compute  $h_n$ ,  $q_n$ ,  $D_n$ ,  $B_n$ ,  $C_n$ ,  $q_{n+1}$  as defined previously.
  - 2. Compute the next approximation  $X_{n+1}$  by

$$X_{n+1} = X_n + h_n q_{n+1}$$
.

3. Test for convergence; that is, test

$$|X_{n+1} - X_n| / |X_{n+1}| < \epsilon$$

for some suitably small number  $\varepsilon$ .

4. If the test fails, return to step 1 with the last three approximations  $X_{n+1}$ ,  $X_n$ ,  $X_{n-1}$ .

- 5. If the test passes, do the following:
  - a. Save  $X_{n+1}$  as the desired approximation to a zero.
  - b. Deflate the current polynomial using  $X_{n+1}$ .
  - c. Replace the current polynomial by the deflated polynomial.
  - d. Return to step 1 with a new set of initial approximations.

In order to avoid an unending iteration process in case the method does not produce convergence, a maximum number of iterations should be specified. If convergence is not obtained within this number of iterations, the initial approximations should be altered.

3. Geometrical Interpretation of Muller's Method

Figure 3.1. shows the geometrical interpretation of Muller's method for real roots of P(X) and the quadratic Q(X). The root of Q(X) closest to  $X_i$  is chosen as the next approximation  $X_{i+1}$ .

4. Determining Multiple Roots

For a discussion concerning multiple roots see Chapter II, § 5.

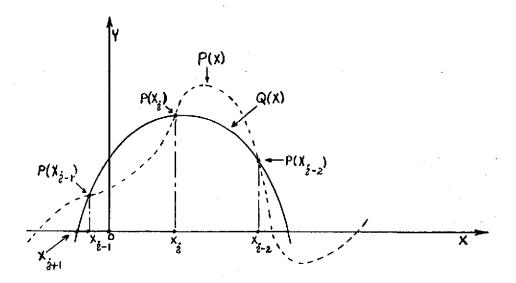


Figure 3.1. Geometrical Interpretation of Muller's Method

#### CHAPTER IV

#### GREATEST COMMON DIVISOR METHOD

## 1. Derivation of the Algorithm

The greatest common divisor (g.c.d.) method reduces the problem of finding all the zeros of a polynomial, possibly having multiple zeros, to one of solving for zeros of a polynomial all of whose zeros are simple.

Consider the N<sup>th</sup> degree polynomial

$$P(X) = a_1 X^N + a_2 X^{N-1} + ... + a_N X + a_{N+1}$$

where  $a_1 \neq 0$  and  $a_1, a_2, \dots, a_{N+1}$  are complex numbers. If P(X) has m distinct zeros,  $\alpha_1, \alpha_2, \dots, \alpha_m$ , then P(X) can be expressed in the form

$$P(X) = a_1(X - \alpha_1)^{e_1} (X - \alpha_2)^{e_2} ... (X - \alpha_m)^{e_m}$$
 (4-1)

where  $e_i$  is the multiplicity of  $\alpha_i$ ,  $i=1,2,\ldots,m$ . The derivative of P(X) is

$$P'(X) = N a_1 X^{N-1} + (N-1) a_2 X^{N-2} + ... + a_N$$

which can also be expressed as

$$P'(X) = a_{1}(X - \alpha_{i})^{e_{1}-1} (X - \alpha_{2})^{e_{2}-1} \dots (X - \alpha_{m})^{e_{m}-1} \sum_{i=1}^{m} e_{i} \prod_{\substack{j=1 \ j \neq i}}^{m} (X - \alpha_{j}).$$
(4-2)

The greatest common divisor of P(X) and P(X) is obtained from the following theorem.

Theorem 4.1. Let P(X) be an  $N^{th}$  degree polynomial having m distinct zeros  $\alpha_1, \alpha_2, \ldots, \alpha_m$  of multiplicity  $e_1, e_2, \ldots, e_m$  respectively. Then the polynomial

$$D(X) = (X - \alpha_1)^{e_1^{-1}} (X - \alpha_2)^{e_2^{-1}} ... (X - \alpha_m)^{e_m^{-1}}$$

is the unique monic greatest common divisor of P(X) and its derivative P'(X).

<u>Proof.</u> Since the set of all polynomials over the complex number field is a unique factorization domain and since each factor  $X - \alpha_i$  is irreducible, it follows from (4-1) and (4-2) that D(X) is the unique monic greatest common divisor of P(X) and P'(X).

It follows from Theorem 4.1 that each zero of D(X) is also a zero of P(X) and P'(X). Hence we have the following result.

Theorem 4.2. If P(X) is a polynomial, then P(X) and P'(X) are relatively prime if and only if P(X) has no multiple zeros.

Consider the polynomial H(X) obtained by dividing P(X) by its monic g.c.d., D(X).

$$H(X) = P(X)/D(X)$$

$$= a_1 \underbrace{\prod_{i=1}^{m} (X - \alpha_i)^{e_i}}_{i=1} / \underbrace{\prod_{i=1}^{m} (X - \alpha_i)^{e_i-1}}_{i=1}$$

$$= a_1 \prod_{i=1}^{m} (X - \alpha_i).$$

The zeros of H(X) are all simple zeros and are also all the distinct zeros of P(X). Use of the g.c.d. method involves computation of H(X) when given P(X).

In order to obtain H(X), a computational algorithm is necessary to find the g.c.d. of P(X) and P'(X). The general method for computing the g.c.d. of two polynomials is as follows: Let  $R_0(X)$  and  $R_1(X)$  be two polynomials having degrees  $N_0$  and  $N_1$  respectively such that  $N_1 \leq N_0$ . The g.c.d. of  $R_0(X)$  and  $R_1(X)$  is desired. By the division algorithm, there exists polynomials  $S_1(X)$  and  $R_2(X)$  such that

$$R_0(X) = R_1(X) S_1(X) + R_2(X)$$

where either  $R_2(X) = 0$  or deg.  $R_2(X) < deg.$   $R_1(X)$ . Similarly if  $R_2(X) \neq 0$ , there exists polynomials  $S_2(X)$  and  $R_3(X)$  such that

$$R_1(X) = S_2(X) R_2(X) + R_3(X)$$

where either  $R_3(X) = 0$  or deg.  $R_3(X) < \deg$ .  $R_2(X)$ . Continuing in the above manner, suppose  $R_1(X)$  and  $R_{i+1}(X)$  have been found where deg.  $R_{i+1}(X) < \deg$ .  $R_1(X)$ . Then there exists polynomials  $R_{i+2}(X)$  and  $R_{i+1}(X)$  such that

$$R_{i}(X) = R_{i+1}(X) S_{i+1}(X) + R_{i+2}(X)$$

where either  $R_{i+2}(X) = 0$  or deg.  $R_{i+2}(X) < \deg$ .  $R_{i+1}(X)$ . Then we obtain a sequence  $R_0(X)$ ,  $R_1(X)$ ,...,  $R_K(X)$ ,  $R_{K+1}(X)$  such that deg.  $R_i(X) < \deg$ .  $R_{i-1}(X)$ , i = 1, 2, ..., K+1. Since a polynomial cannot have degree less than zero, the above process, in a finite number of steps (at most  $N_1$ ), results in polynomials  $R_{K-1}(X)$ ,  $S_K(X)$  and  $R_K(X)$  with deg.  $R_K(X) < \deg$ .  $R_{K-1}(X)$  such that

$$R_{K-1}(X) = R_K(X) S_K(X) + R_{K+1}(X)$$

and  $R_{K+I}(X) = 0$ .

Theorem 4.3. Let the sequence  $R_0(X)$ ,  $R_1(X)$ ,...,  $R_K(X)$ ,  $R_{K+1}(X)$  be defined as above. Then  $R_K(X)$  is the greatest common divisor of  $R_0(X)$  and  $R_1(X)$ .

<u>Proof.</u> It is clear that  $R_K(X)$  divides  $R_{K-1}(X)$ . If  $R_K(X)$  divides  $R_i(X)$  for  $0 \le i \le k$ , then  $R_j(X) = R_{j+1}(X)$   $S_{j+1}(X) + R_{j+2}(X)$ . Thus,  $R_K(X)$  divides  $R_j(X)$  and it follows by induction that  $R_K(X)$  divides both  $R_0(X)$  and  $R_1(X)$ . By reversing the inductive argument given above, it is easy to see that if L(X) divides  $R_0(X)$  and  $R_1(X)$ , the L(X) divides  $R_i(X)$  for  $i = 0, 1, \ldots, K$ . Therefore, L(X) divides  $R_K(X)$  which shows that  $R_K(X)$  is the greatest common divisor of  $R_0(X)$  and  $R_1(X)$ .

The above theorem tells how to obtain the greatest common divisor of two polynomials. A machine oriented method is now developed for computing the sequence of  $R_j(X)$ 's. Beginning the sequence with  $R_0(X)$  and  $R_1(X)$ , the polynomial  $R_{i+1}(X)$  of the sequence is derived from  $R_i(X)$ 

and  $R_{i-1}(X)$  as follows: Let  $R_{i-1}(X)$  of degree  $N_{i-1}$  be given by

$$R_{i-1}(X)$$
=  $r_{i-1,1} X^{N_{i-1}} + r_{i-1,2} X^{N_{i-1}-1} + \dots + r_{i-1,N_{i-1}} X + r_{i-1,N_{i-1}+1}$ 

and  $R_{i}(X)$  of degree  $N_{i}$  be given by

$$R_{i}(X) = r_{i,1} X^{i} + r_{i,2} X^{i-1} + ... + r_{i,N_{i}} X + r_{i,N_{i}+1}$$

where  $N_i \leq N_{i-1}$ . Define  $U_1(X)$  by

$$U_1(X) = (r_{i-1,1}/r_{i,1}) X^{N_{i-1}-N_{i}}$$
.

Then define  $T_1(X)$  by

$$T_{1}(X) = R_{i-1}(X) - U_{1}(X) R_{i}(X)$$

$$= [r_{i-1,1} - r_{i,1} (r_{i-1,1} / r_{i,1})] X^{N_{i-1}}$$

$$+ [r_{i-1,2} - r_{i,2} (r_{i-1,1} / r_{i,1})] X^{N_{i-1}-1}$$

$$+ ...$$

$$+ [r_{i-1,N_{i-1}+1} - r_{i,N_{i-1}+1} (r_{i-1,1} / r_{i,1})]$$

where  $r_{i,j} = 0$  for  $j > N_i + 1$ .

We consider three cases.

- (1) If  $T_1(X) = 0$ , then  $R_1(X) = R_K(X)$ ; that is,  $R_1(X)$  is the g.c.d. of  $R_0(X)$  and  $R_1(X)$ .
- (2) If  $T_1(X) \neq 0$  and deg.  $T_1(X) < N_1$ , then  $R_{i+1}(X) = T_1(X)$ .

(3) If  $T_1(X) \neq 0$  and deg.  $T_1(X) = M_1 \geq N_1$ , then define  $U_2(X)$  by

$$U_2(X) = (t_{1,1}/r_{1,1}) X^{M_1-N_1}$$

where

$$T_1(X) = t_{1,1} X^{M_1} + t_{1,2} X^{M_1-1} + \dots + t_{1,M_1} X + t_{1,M_1+1}.$$

Define  $T_2(X) = T_1(X) - U_2(X) R_1(X)$  which can be expressed by

$$T_{2}(X) = [t_{1,1} - (t_{1,1}/r_{1,1}) r_{1,1}] X^{M_{1}-1}$$

$$+ [t_{1,2} - (t_{1,1}/r_{1,1}) r_{1,2}] X^{M_{1}-2}$$

$$+ ...$$

$$+ [t_{1,M_{1}+1} - (t_{1,1}/r_{1,1}) r_{1,M_{1}+1}]$$

where  $r_{i,j} = 0$  for  $j > N_i+1$ . We again consider the following three cases.

- (1) If  $T_2(X) = 0$ , then  $R_1(X)$  is the g.c.d. of  $R_0(X)$  and  $R_1(X)$ .
- (2) If  $T_2(X) \neq 0$  and deg.  $T_2(X) < \deg$ .  $R_1(X)$ , then  $R_{i+1}(X) = T_2(X)$ .
- (3) If  $T_2(X) \neq 0$  and deg.  $T_2(X) = M_2 \geq N_1$ , then define  $U_3(X)$  by

$$U_3(X) = (r_{2,1}/r_{1,1}) x^{M_2-N_1}$$

where

$$T_2(X) = t_{2,1} X^{M_2} + t_{2,2} X^{M_2-1} + \dots + t_{2,M_2} X + t_{2,M_2+1}$$

Since deg.  $T_{i+1}(X) < deg.$   $T_{i}(X)$ , then this process is finite (not to exceed  $N_{i-1}$ ) ending, for some integer S, in  $T_{S}(X)$  such that

- (1)  $T_S(X) = 0$  and  $R_1(X)$  is the g.c.d. of  $R_0(X)$  and  $R_1(X)$  or
- (2)  $T_S(X) \neq 0$  but deg.  $T_S(X) < \deg$ .  $R_1(X)$ , in which case  $T_S(X) = R_{i+1}(X)$ .

Thus, using this algorithm and given  $R_0(X)$  and  $R_1(X)$ , the sequence  $R_0(X)$ ,  $R_1(X)$ ,  $R_2(X)$ ,...,  $R_i(X)$ ,  $R_{i+1}(X)$  can be generated such that either

- (1)  $R_{i+1}(X) = 0$  and  $R_i(X)$  is the g.c.d. of  $R_0(X)$  and  $R_1(X)$  or
- (2)  $R_{i+1}(X) \neq 0$  and  $N_{i+1} \leq N_{i}$ . In a finite number of iterations,  $R_{K}(X)$ , the g.c.d. of  $R_{0}(X)$  and  $R_{1}(X)$ , can be obtained.

Recall that we wanted to obtain the polynomial H(X) = P(X)/D(X) where D(X) is the g.c.d. of P(X) and P'(X). Thus, after obtaining D(X) by the above algorithm, it is necessary to divide P(X) by D(X) obtaining H(X) all whose zeros are simple.

Once H(X) is obtained, an appropriate method such as Newton's method or Muller's method is applied to extract the zeros of H(X). This gives all the zeros of P(X).

As in Newton's or Muller's method, the zeros may be checked for accuracy and possibly improved by using them as initial approximations with the particular method applied to the full (undeflated) polynomial, P(X).

# 2. Determining Multiplicities

After all zeros of P(X) are found, the multiplicity of each zero can be determined by the process outlined in Chapter II, § 5.

# 3. Procedure for the G.C.D. Method

The basic steps performed by the greatest common divisor method are listed sequentially as follows:

- 1. Given a polynomial, P(X), in the form  $P(X) = a_1 X^{N} + a_2 X^{N-1} + \dots + a_N X + a_{N+1}.$
- 2. Calculate the derivative, P'(X), of P(X) in the form  $P'(X) = b_1 X^{N-1} + b_2 X^{N-2} + \dots + b_N \text{ where } b_1 = Na_1,$   $b_2 = (N-1)a_2, \dots, b_N = a_N.$
- 3. Find D(X), the g.c.d. of P(X) and P'(X) using the algorithms developed above.
- 4. Calculate H(X) = P(X)/D(X), the polynomial having only simple zeros.
- 5. Use some appropriate method to extract the zeros of H(X).
- 6. Determine the multiplicity of each of the zeros obtained in step 5.

# CHAPTER V

## REPEATED GREATEST COMMON DIVISOR METHOD

# 1. Derivation of the Algorithm

The repeated greatest common divisor (repeated g.c.d.) method makes repeated use of the g.c.d. method to extract the zeros and their multiplicatives of a polynomial with complex coefficients. That is, the repeated g.c.d. method reduces the problem of finding the zeros of a polynomial, P(X), which possibly has multiple zeros, to one of finding the zeros of a polynomial which has only simple zeros and the zeros of this polynomial are all the zeros of P(X) of a given multiplicity.

Let

$$P(X) = a_1 X^{N} + a_2 X^{N-1} + \dots + a_N X + a_{N+1}$$

$$= a_1 (X - \alpha_1)^{e_1} (X - \alpha_2)^{e_2} \dots (X - \alpha_m)^{e_m}$$

where  $a_1 \neq 0$ , each  $a_i$  is a complex number, and  $\alpha_1, \alpha_2, \dots, \alpha_m$  are the distinct zeros of P(X) having multiplicity  $e_1, e_2, \dots, e_m$ , respectively. If  $D_1(X)$  is the monic greatest common divisor of P(X) and P'(X), then Theorem 4.1 shows that

$$D_1(X) = (X - \alpha_1)^{e_1-1} (X - \alpha_2)^{e_2-1} ... (X - \alpha_m)^{e_m-1}$$

where we assume that if  $e_j$  = 1, then X -  $\alpha_j$  does not appear in the

representation. Let  $D_2(X)$  be the monic greatest common divisor of  $D_1(X)$  and  $D_1(X)$ . Then

$$D_2(X) = (X - \alpha_1)^{e_1-2} (X - \alpha_2)^{e_2-2} ... (X - \alpha_m)^{e_m-2}$$

where we assume that if  $e_j \leq 2$ , then  $X - \alpha_j$  does not appear in the representation. From the above it is clear that the zeros of  $D_1(X)$  are just the multiple zeros of P(X) to one lower power. The zeros of  $D_2(X)$  are just the multiple zeros of  $D_1(X)$  to one lower power. Thus, the zeros of  $D_2(X)$  are just the zeros of P(X) which have multiplicity greater than two, and their multiplicity in  $D_2(X)$  is reduced by two. Therefore, it follows that

$$G_1(X) = [P(X)/D_1(X)]/[D_1(X)/D_2(X)] = P(X)D_2(X)/[D_1(X)]^2$$

has only simple zeros and they are just the simple zeros of P(X). In general if  $D_{\mathbf{j}}(X)$  has been defined for  $1 \leq \mathbf{j} \leq \mathbf{i}$  and if  $D_{\mathbf{j}+1}(X)$  is the monic greatest common divisor of  $D_{\mathbf{j}}(X)$  and  $D_{\mathbf{j}}(X)$ , then the zeros of  $D_{\mathbf{j}+1}(X)$  are the multiple zeros of  $D_{\mathbf{j}}(X)$  to one lower power. Thus, the zeros of  $D_{\mathbf{j}+1}(X)$  are just the zeros of P(X) which have multiplicity greater than  $\mathbf{i}+1$  and their multiplicity in  $D_{\mathbf{j}+1}(X)$  is reduced by  $\mathbf{i}+1$ . It follows that

$$G_{i}(x) = [D_{i-1}(x)/D_{i}(x)]/[D_{i}(x)/D_{i+1}(x)]$$
  
=  $D_{i-1}(x) D_{i+1}(x)/[D_{i}(x)]^{2}$ 

has simple zeros and they are just the zeros of P(X) that have multiplicity i. Thus, we have proven the following theorem.

Theorem 5.1. Let  $P(X) = a_1 X^N + a_2 X^{N-1} + \dots + a_N X + a_{N+1}$  where  $a_1 \neq 0$  and  $a_1, a_2, \dots, a_{N+1}$  are complex numbers. If  $D_0(X) = P(X)$  and if  $D_{i+1}(X)$  is the monic greatest common divisor of  $D_i(X)$  and  $D_i(X)$  for  $i \geq 0$ , then

$$G_{i}(X) = D_{i-1}(X) D_{i+1}(X) / [D_{i}(X)]^{2}$$

has only simple zeros and they are just the zeros of P(X) that have multiplicity i.

Thus, by the above theorem we can generate a sequence of polynomials  $G_1(X)$ ,  $G_2(X)$ ,...,  $G_K(X)$  where the set of zeros of P(X) is the same as the set of zeros of this sequence and the multiplicity of each zero in P(X) is given by the corresponding subscript on G(X). Therefore, by using a method such as Newton's method of Muller's method to calculate the zeros of each  $G_1(X)$ , we will have the zeros of P(X) along with their multiplicities.

2. Procedure for the Repeated G.C.D. Method

The basic steps performed by the greatest common divisor method are listed sequentially as follows:

- 1. Given a polynomial, P(X), in the form  $P(X) = a_1 X^N + a_2 X^{N-1} + \dots + a_N X + a_{N+1}.$
- 2. Set  $D_0(X) = P(X)$ .
- 3. Calculate the derivative,  $D_0'(X)$ , of  $D_0(X)$  in the form

$$D_0'(X) = b_1 X^{M-1} + b_2 X^{M-2} + \dots + b_M$$

where deg. 
$$D_0(X) = M$$
,  $D_0(X) = d_1 X^M + ... + d_{M+1}$ ,  
and  $b_1 = Md_1$ ,  $b_2 = (M-1)d_2$ , ...,  $b_M = d_M$ .

- 4. Find  $D_1(X)$ , the g.c.d. of  $D_0(X)$  and  $D_0'(X)$  using the algorithms developed in Chapter IV.
- 5. Similar to 3., calculate  $D_1(X)$ .
- 6. Find  $D_2(X)$ , the g.c.d. of  $D_1(X)$  and  $D_1(X)$  using the algorithms developed in Chapter IV.
- 7. Calculate  $G(X) = D_0(X) D_2(X)/[D_1(X)]^2$ .
- 8. Use some appropriate method to extract the zeros of G(X) and assign these zeros the correct multiplicity as zeros of P(X).
- 9. Set  $D_0(X) = D_1(X)$ ,  $D_0'(X) = D_1'(X)$ , and  $D_1(X) = D_2(X)$ . Then repeat 5.-8. above until all the zeros of P(X) are found.

# CHAPTER VI

## CONCLUSION

In order to compare Newton's, Muller's, the greatest common divisor, and the repeated greatest common divisor methods, we consider the polynomials as being divided into the following classes:

- 1. polynomials with all distinct zeros.
- 2. polynomials with multiple zeros.

The comparisons in the following material are results of tests made on the IBM 360/50 computer which has a 32 bit word. The programs were successfully run on the CDC 6600 and the UNIVAC 1108 which have a 60 bit word and a 36 bit word respectively. It was noted that the UNIVAC 1108 is about 15 times faster than the IBM 360/50. The CDC 6600 is faster than the UNIVAC 1108 but the difference is not as great as that between the UNIVAC 1108 and the IBM 360/50.

# 1. Polynomials With all Distinct Zeros

First we consider the class of polynomials having distinct zeros. Newton's method is particularly suited for this class of polynomials. Its quadratic convergence is very fast which can save time and money to the user. The accuracy obtained is excellent as shown in Exhibit 6.1 which presents the zeros of a 15<sup>th</sup> degree polynomial in double precision. In most cases, the method produces convergence for almost any initial approximation given.

Muller's method also produces good results on this class of polynomials. The rate of convergence is, however, somewhat slower than Newton's method. This fact is especially significant when working with polynomials of high degree. The accuracy obtained by Muller's method is comparable to, but does not exceed that of Newton's method. In most cases, the accuracy of the two methods does not differ by more than one or two decimal places. Exhibit 6.2 shows results of Muller's method for the polynomial of Exhibit 6.1. As in Newton's method, convergence is produced for almost any initial approximation given.

The g.c.d. method, whether used with Newton's or Muller's method as a supporting method on this class of polynomials, is no better than Newton's or Muller's method alone. The reason for this is that the greatest common divisor of the polynomial, P(X), and its derivative is 1. Thus,  $H(X) \approx P(X)/g.c.d.$  P(X) = P(X); that is, the polynomial solved by the supporting method is the same as the original polynomial. Thus, in this case the g.c.d. method will not produce better results than the supporting method used alone. The above comments also hold for the repeated g.c.d. method.

Thus, this class of polynomials presents no difficulty to any of these four methods. Newton's method, because of its speed, is therefore recommended.

# 2. Polynomials With Multiple Zeros

Next consider the class of polynomials containing multiple zeros. Exhibits 6.3 - 6.26 illustrate output from six different programs using the methods described in Chapters II - V. Four polynomials are used where the zeros of these polynomials are listed below. The number in

parentheses indicates the multiplicity of that zero.

Polynomial #1	Polynomial #2	Polynomial #3	Polynomial #4
2+2i (3) 1+2i (2) -1+.5i (1)	-2.33 (1) .003 (2) i (2) 1.5i (2) -1.5i (2) 3i (3) -1-i (3)	2+2i (3) 1+2i (2) -1+5i (3)	1+i (6) 1-i (6)

\*\*\*\*\*\*\*\*\*

Note the relationship between polynomials #1 and #3.

This class presents considerable difficulty for Newton's method, especially those polynomials containing zeros of high multiplicity or containing a considerable number of multiple zeros. The iteration formula for Newton's method is

$$X_{n+1} = X_n - P(X_n)/P'(X_n).$$

If c is a multiple zero then P(c) = P'(c) = 0. Hence, as  $X_n \to c$ ,  $P(X_n) \to 0$  and  $P'(X_n) \to 0$  and the iteration formula may be unstable, resulting in no convergence or bad accuracy. As the number of multiple zeros increases, the polynomial becomes more ill-conditioned, convergence becomes more difficult, and accuracy is lost. Thus, the possibility of convergence decreases. This also holds true if the multiplicities of the zeros are increased. The rate of convergence of Newton's method is much slower for multiple zeros than for distinct zeros. Exhibit 6.3 shows a polynomial (#1) containing two multiple zeros solved in double precision. Note the following from Exhibit 6.3.

 Roots #2 and #3 are greatly improved by iterating on the original polynomial. Distinct roots are usually improved in this manner.

- 2. The time taken to solve this 6<sup>th</sup> degree equation with multiple roots is greater than the time taken by the same program to solve a 15<sup>th</sup> degree polynomial with all distinct roots (Exhibit 6.1).
- 3. Root #2 did not pass the convergence test after 200 iterations even though it was improved. This is probably due to the fact that the polynomial from which root 2 was extracted had only one multiple root but the original polynomial from which it was extracted the second time had two multiple roots; that is, the original polynomial is more ill-conditioned.
- 4. The accuracy of the roots before the attempt to improve accuracy is very poor. Root #2 is accurate to only three decimal places as compared to the 15 decimal places in Exhibit 6.1 for distinct roots.

  Root #3 is especially bad, the imaginary part being accurate to only one decimal place.

Exhibit 6.4 uses polynomial #2. Note the poor results obtained before the attempt to improve accuracy and the improvement afterward. Also note that after the attempt to improve accuracy, one of the zeros, namely 3i, is lost and an extra zero, namely 1.5, is included in the list. (See Appendix A, § 4.) A convergence requirement of 10<sup>-5</sup> was used on this polynomial to get it to converge to all of the zeros in a maximum number of 200 iterations.

In many cases, Newton's method fails to converge altogether.

Polynomial #3 could not be solved using Newton's method with a maximum

number of 200 iterations and a convergence requirement of  $10^{-9}$ . Exhibit 6.5 illustrates the bad results for a convergence requirement of  $10^{-5}$  which was needed in order to get convergence. In Exhibit 6.6 a convergence requirement of  $10^{-3}$  was needed in order to get convergence to the zeros of polynomial #4.

Muller's method also encounters difficulty, although to a lesser degree than Newton's method, on this class of polynomials. In most cases, Muller's method produces convergence even when Newton's method completely failed for polynomials #3 and #4 with a convergence requirement of 10<sup>-9</sup> but convergence was obtained using Muller's method as shown in Exhibits 6.9 and 6.10. The accuracy obtained by Muller's method is not good but usually better than Newton's method using the same convergence requirement. The rate of convergence of Muller's method is considerably slower for multiple zeros than for distinct zeros. However, for multiple zeros, Muller's method is as fast or faster than Newton's method is considerably slower for multiple zeros, Muller's method is as fast or faster than Newton's method is not good but usually better

The g.c.d. method is perfectly suited for polynomials with multiple zeros. All multiple zeros are removed leaving only a polynomial of class 1 (all distinct roots) to be solved. This indicates that best results should be obtained by using Newton's method as the supporting method, since Newton's method enjoys the advantage of speed over Muller's method for distinct zeros. This has indeed proved to be true. The accuracy of the zeros obtained decreases, somewhat, when the number of multiple zeros is increased. This is due to accuracy lost in computing the g.c.d. and the quotient polynomial and not as a result of the supporting method. It is easy to see that the accuracy of the g.c.d. method is best when the degree of the greatest common divisor of

P(X) and P'(X) is maximum. This is due to the fact that the error in the greatest common divisor is minimized in this case. The accuracy obtained using Newton's method and Muller's method as supporting methods is about the same. This is verified by Exhibits 6.11 - 6.14 (g.c.d. method with Newton) and Exhibits 6.15 - 6.18 (g.c.d. method with Muller).

Multiplicities are determined with excellent accuracy. The g.c.d. method is not as sensitive to zeros of high multiplicity or polynomials containing a large number of multiple zeros as are both Newton's and Muller's methods. A quick comparison of Exhibits 6.11 - 6.14 and 6.15 - 6.18 with Exhibits 6.3 - 6.6 and 6.7 - 6.10 show that the g.c.d. method with either supporting method is much more accurate than either Newton's or Muller's method. For example, Exhibits 6.5 and 6.9 show polynomial #3 for which Newton's method and Muller's method both gave poor convergence. But Exhibits 6.13 and 6.17 show very accurate results for polynomial #3.

The repeated g.c.d. method is also suited very well for polynomials with multiple zeros. Exhibits 6.19 - 6.22 and Exhibits 6.23 - 6.26 are results of the repeated g.c.d. method with Newton's method and Muller's method as supporting methods, respectively. However, the results of the repeated g.c.d. method are not as good as those obtained from the g.c.d. method. Since the repeated g.c.d. method repeated uses the g.c.d. algorithm, the error tends to build up in this method when a polynomial has several zeros of different multiplicities. This can be observed by comparing Exhibits 6.20 and 6.24 with Exhibits 6.12 and 6.16 on polynomial #2 and by comparing Exhibits 6.21 and 6.25 with Exhibits 6.13 and 6.17 on polynomial #3. As was the case of the g.c.d.

method, there is little difference between the repeated g.c.d. method with Newton's method or Muller's method as a supporting method. This can be observed by comparing Exhibits 6.19 - 6.22 (Newton) with Exhibits 6.23 - 6.26 (Muller). Even though the results of the repeated g.c.d. method are not quite as good as the results of the g.c.d. method, they are far superior to the results of both Newton's method and Muller's method.

Table 6.I gives a comparison of the execution times of the six methods for polynomials #1 - #4.

# TABLE 6.I

- ----

METHOD	EXECUTION TIME
Newton	104.16 seconds
Muller	96.79 seconds
G.C.D. with Newton	7.51 seconds
G.C.D. with Muller	8.91 seconds
Repeated G.C.D. with Newton	7.71 seconds
Repeated G.C.D. with Muller	15.16 seconds

It is clear from Table 6.I that the g.c.d. and the repeated g.c.d. methods are much faster than both Newton's and Muller's method on

 $<sup>^{*}</sup>$  These times are from execution runs on the IBM 360/50 WATFOR system.

polynomials with multiple zeros. Therefore, for polynomials with multiple zeros, the order in which the methods are recommended is as follows.

- 1. G.C.D. with Newton.
- 2. G.C.D. with Muller.
- 3. Repeated G.C.D. with Newton.
- 4. Repeated G.C.D. with Muller.
- 5. Muller.
- 6. Newton.

# NEWTONS METHOD TO FIND ZEROS OF POLYMONIALS POLYMONIAL NUMBER 7 OF DEGREE 15

## THE COEFFICIENTS OF P(X) ARE

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.100-09
TEST FOR MULTIPLICITIES. 0.100-01
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

BEFORE THE ATTEMPT TO IMPROVE ACCURACY. THE ZEROS OF PIX) ARE

ROOTS OF P(X)	•	MULTIPLICITIES	INITIAL	APPROXIMATION
ROUT( 1) = 0.30000000000000000000000000000000000	• 0.1609358115166531D-16 1	1	0.4829629115656279D 00	+ 0.12940952844381870 00 1
RODE( 2) + 0.2005000000000000000000000000000000000	-0.2000000000000000 00 T	ı	0.7071067553046346D 00	
RDDT( 3) = -0.200000000000000000000000000000000000		1	0.3882284792654056D 00	
#DOT: 41 = 0.2485026109655803D-17	• 0.100000000000000000000000000000000000		-0.5176382551966724D 00	
RDDT1 = -0.100000000000000000000000000000000000	+ -0.5321953706459946D-16 I		-0.1767767147080701D 01	
RODT( 6) = -0.100000000000000000000000000000000000	+ -0.99999999999999BD DD			+ 0.7764567463987070D 00 I
RDD14 71 = -0.31069187592803490-15	• -0.99999999999994D			+ -0.9058671940B16160D DO 1
RODT( 8) = -D.200000000000000000 01	• -0.30000000000000000 01 L			+ -0.2828427642384390D 01 l
900T( 9) = 0.10000000000000160 01	<ul> <li>-0.1009764996352100D-12 1</li> </ul>	ì		+ -0.4346666459873368D 01 I
R30T(10) = 0.20000000000000000000000000000000000	+ -B.1000000000000032D 01 1	1		+ -0.4829628831453027D D1 1
#ORT(111 = 0.19999999999999800 01 ·	<ul> <li>0.1385507847222202D-12 1</li> </ul>	1		+ -0.3889086300072258D 01 I
RDDT(12) - 0.30000000000000000000000000000000000	+ -0.1417549157449219D-13 I	1		+ -0.1552912644268974D 01 1
RODT(13) = 0.3999999999999980 01	• 0.400000000000000000000000000000000000	1		+ 0.1682325708247752D OL I
RODT(14) = 0.99999999999996060 00	+ 0.1000000000000005D 01 I	1	SOLVED BY DIRECT METHOD	
ROOT(15) = -0.3333333333333333340 01	<ul> <li>-0.1850371707708594D-15 1</li> </ul>	. <b>1</b>	SOLVED BY DIRECT METHOD	l

Exhibit 6.1.

AFTER THE ATTEMPT TO IMPROVE ACCURACY. THE ZEROS OF PIXT ARE

Exhibit 6.1, Roots Are: -1 - i, 1 + i, -2 - 3i, 2 - i, 3, 2, i, -i, -10/3, .3, -1, 1, 4 + 4i, -.2 + .2i, .2 - .2i.

MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL POLYNOMIAL NUMBER 7 OF DEGREE 15

#### THE COEFFICIENTS OF P(X) ARE

```
        P( 1) = 0.30000000000000000 01 + 0.000000000000000 00 0 1

        P( 2) = 0.179000000000000 02 + 0.000000000000000 00 1

        P( 3) = 0.2010000000000000 02 + 0.000000000000000 00 1

        P( 4) = 0.1745000000000000 03 + 0.2843680000000000 03 1

        P( 5) = 0.759420000000000 03 + 0.3118880000000000 03 1

        P( 6) = 0.8274360000000000 03 + 0.3118880000000000 04 1

        P( 7) = 0.1327984000000000 04 + 0.41102400000000 04 1

        P( 8) = 0.5611860000000000 04 + 0.41102400000000 04 1

        P( 9) = 0.7224756000000000 04 + 0.1548288000000000 04 1

        P( 10) = 0.227699200000000 04 + 0.30463200000000 04 1

        P( 11) = 0.540280000000000 04 + 0.30463200000000 04 1

        P( 12) = 0.540280000000000 04 + 0.3263880000000000 04 1

        P( 13) = 0.04683800000000000 04 + 0.3263880000000000 04 1

        P( 14) = 0.16474560000000000 04 + 0.326880000000000 04 1

        P( 13) = 0.4668360000000000 04 + 0.326880000000000 02 1

        P( 13) = 0.466800000000000 03 - 0.3475840000000000 03 1
```

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.100-09
TEST FOR MULTIPLICITIES. 0.100-01
RADIUS TO START SERRCH. 0.00D 00
RADIUS TO END SEARCH. 0.00D 00

BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF PEXT		MULTIPLICITIES	INITIAL APPROXIMATION
	•		
ROOT( 1) = 0.30000000000000000000000000000000000	98751751866D-16 I	1	D.4829629115656279D OC + D.1294D9528443B187D DC I
RDD1 ( 2) = 0.20000000000000000 00 + -0.20000	1 00 00000000000 00 1	1	0.70710675530463460 00 * 0.70710680706845950 00 1
ROST( 3) = 0.9999999999999950 CO + 0.99999	9999999997D 00 1	1	0.3882284792654056D CO + 0.1448888763117193D OI I
RDDT( 4) = -0.2389082408313382D-16 + D.10000	000000000000 01 I	1	-0.5176382551966724D 00 + 0.1931851608368755D 01 1
RD3T( 5) * -0.100000000000000000 01 + 0.25316	29277617640D-15 I	ı	-0.17677671470807010 01 + 0.17677667588520150 01 1
ROOT ( $61 = -0.33333333333333340 01 + -0.21447$	226690877370-15 1	1	-0.28977775830749900 01 + 0.77645674639870700 00 1
RDDT(T) = -0.10000000000000000000 + -0.99999		1	-0.3380740248331229D 01 + -0.9058671940816160D 00 [
ROOT( 8) = -0.2000000000000000000 D1 + -0.30000		1	-0.28284266071078969 01 + -0.28284276423843900 01 I
	00000000109D 01 I	i	-0.11646848013998990 01 + -0.43466664598733680 01 1
RADT(10) = 0.199999999999879D 01 + -0.99999	99999999954D 00 I	1	0.1294096345098645D 01 + -0.4829628831453027D 01
	912044391240-14 1	ı	0.3889088292979509D 01 + -0.3889086300072258D 01 1
	54879580170D-12 !	. 1	0.5795555393512303D 01 + -0.1552912644268974D 01 1
	0000000000050 01 1	. 1	0.62785173577341250 01 + 0.16823257082477520 01 1
	44641937337D-12 I	1	SOLVED BY DIRECT METHOD
	D00000000595D 00 I	1	SOLVED BY DIRECT METHOD

Exhibit 6.2.

-AFTER THE ATTEMPT TO EMPROVE ACCURACY

Exhibit 6.2. Roots Are: -1 - i, 1 + i, -2 - 3i, 2 - i, 3, 2, i, -i, -10/3, .3, -1, 1, 4 + 4i, -.2 + .2i, .2 - .2i.

# MENTONS METHOD TO FIND ZEROS OF POLYNOMIALS POLYNOMIAL NUMBER 1 OF DEGREE 6

```
THE COEFFICIENTS OF PIXI ARE
 Same and the second of the second second of the second
NUMBER OF INITIAL APPROXIMATIONS GIVEN.
                               200
MAXIMUM NUMBER OF ITERATIONS.
TEST FOR CONVERGENCE.
                            0.100-09
TEST FOR MULTIPLICITIES.
                            0.100-01
RADIUS TO START SEARCH.
                            0.000 00
                            0.000 00
RADIUS TO END SEARCH.
BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF PIX) ARE
 ROOTS OF P(X)
```

```
RODT( 1) = 0.9999998836125019D 00 + 0.2000000052138284D 01 1 2 0.4829629115656279D 00 + 0.1294095284438187D 00 I
RODT( 2) = 0.1996737810257486D 01 + 0.1995253821143684D 01 I 3 0.7071067553046346D 00 + 0.7071068070684595D 00 I
RODT( 31 = -0.9902131979974624D 00 + 0.5142384322923812D 00 I 1 SOLVED BY DIRECT METHOD
```

MULTIPLICITIES

INITIAL APPROXIMATION

IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT( 2) \* 0.19967378102574860 01 + 0.19952538211436860 01 1 DID NOT CONVERGE. THE PRESENT APPROXIMATION AFTER 200 ITERATIONS IS PRINTED BELOW.

AFTER THE ATTEMPT TO IMPROVE ACCURACY. THE ZEROS OF PIXT ARE

```
ROOTS OF P1X)

MULTIPLICITIES

INITIAL APPROXIMATION

ROOT( 11 = 0.99999988361250190 00 + 0.2000000052138284D 01 [ 2 0.48296291156562790 00 + 0.1294095284438187D 00 [ ROOT( 21 = 0.19999929075033090 01 + 0.19999594740016890 01 I 3 0.7071067553046346D 00 + 0.7071068070684595D 00 [ ROOT( 3) = -0.99999999999998D 00 + 0.500000000000000000000 00 1 | SQLVED.8Y_DIRECT=METHOD
```

Exhibit 6.3. Roots Are: 2+2i (3), 1+2i (2), -1+.5i

#### NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS POLYNOMIAL NUMBER 2 OF DEGREE 15

#### THE COEFFICIENTS OF PIXT ARE

```
P( 1) = 0.48000000000000000 02 + 0.0000000000000000 00 0 I
P( 2) = 0.25571200000000000 03 + -0.384000000000000000 03 I
P( 3) = -0.7353556800000000 02 + -0.21896960000000000 03 I
P( 4) = -0.3855565896000000 04 + -0.694685145600001 04 I
P( 5) = -0.1733386464800000 05 + -0.1426625972800000 05 I
P( 6) = -0.4967989270400001 05 + -0.1426625972800000 05 I
P( 7) = -0.1023394522130000 06 + -0.603664232000001 04 I
P( 8) = -0.1642742200560000 06 + 0.4137366230400000 05 I
P( 10) = -0.236625888420000 06 + 0.4137366230400000 05 I
P( 11) = -0.1274997298590000 06 + 0.2171341227420000 06 I
P( 11) = -0.1274997298590000 06 + 0.2171341227420000 06 I
P( 11) = -0.3053907774700000 05 + 0.1038130226550000 06 I
P( 11) = -0.1387434434800000 05 + 0.299989414300000 05 I
P( 11) = -0.1387439434434800000 05 + 0.299989414300000 05 I
P( 11) = -0.1387439000000000 05 + 0.299989414300000 05 I
P( 11) = -0.138743990000000000 05 + 0.2998989413000000 05 I
P( 11) = -0.13874344348000000 05 + 0.2998989413000000 05 I
P( 11) = -0.13874344348000000 05 + 0.299889413000000 05 I
P( 11) = -0.13874344348000000 05 + 0.299889413000000 05 I
P( 11) = -0.13874344348000000 05 + 0.299889413000000 05 I
P( 11) = -0.13874344348000000 05 + 0.299889413000000 05 I
P( 11) = -0.13874344348000000 05 + 0.299889413000000 05 I
```

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.100-05
TEST FOR MULTIPLICITIES. 0.100-01
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

BEFORE THE ATTEMPT TO IMPROVE ACCURACY. THE ZEROS OF PIX) ARE

# ROOTS OF P(X) ROOTS (1) = 0.30000007363981840-02 + 0.26093949065529810-08 1 2 0.48296291156562790 00 + 0.12940952844381870 00 1 ROOT( 2) = 0.86266231240991390-04 + 0.10000801737229880 01 1 2 0.388228479265960 00 + 0.707106807068459550 00 1 ROOT( 3) = 0.576634246464275590-02 + 0.15081451743165700 01 1 1 0.38822847926540560 00 + 0.14488887631171930 01 1 ROOT( 4) = -0.59266344710587620-02 + 0.14917229999229610 01 1 1 -0.51763825519667240 00 + 0.19318516083687550 01 1 ROOT( 6) = -0.23333336178348630 01 + 0.1284209313843700 01 1 1 -0.17677671698706040 00 1 ROOT( 7) = -0.10185084355785580 01 + -0.97285956027561890 00 1 1 -0.33807402483312290 01 + 0.90588716408161600 00 1 ROOT( 8) = -0.9654419892828350 00 + -0.9987774789527140 00 1 1 -0.288276767607010 01 + 0.90588716408161600 00 1 1 -0.28827676767607010 01 + 0.9728676664598733680 01 1 ROOT( 9) = -0.24814837099370320-02 + -0.41497879827107940 01 1 2 -0.1164684801399899 01 + -0.43466666598733680 01 1 ROOT( 1) = 0.1560544175373180 00 + 0.24952076852120910 01 1 0.1294093450986450 01 + -0.4829628831453027D 01 1 ROOT( 1) = 0.13256756537607100 00 + 0.336593627402680 01 1 0.1294093450986450 01 + -0.4829628831453027D 01 1 ROOT( 1) = 0.33256756537607100 00 + 0.336593627402680 01 1 0.1294093450986450 01 + -0.4829628831453027D 01 1 ROOT( 1) = 0.43256756537607100 00 + 0.336595687890862D 01 1 1 SOLVED BY DIRECT METHOD

AFTER THE ATTEMPT TO IMPROVE ACCURACY. THE ZEROS OF PIXI ARE

Exhibit 6.4.

Exhibit 6.4. Roots Are: -2.33, .003 (2), i (2), 1.5i (2), -1.5i (2) 3i (3), -1-i (3)

# NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS POLYNOMIAL NUMBER 3 OF DEGREE 8

```
THE COEFFICIENTS OF PIX) ARE
  P( 4) = 0.15725000000000000 03 + 0.1446250000000000 03 1
 P( 5) = 0.3075000000000000 03 + -0.347500000000000 03 I

P( 6) = -0.49525000000000000 03 + -0.49487500000000000 03 I

P( 7) = -0.58575000000000000 03 + 0.49487500000000000 03 I
  Pf 91 = 0.15800000000000000 03 + 0.600000000000000000000 01 f
NUMBER OF INITIAL APPROXIMATIONS GIVEN.
MAXIMUM NUMBER OF ITERATIONS.
                                     200
TEST FOR CONVERGENCE.
                                0-100-05
TEST FOR MULTIPLICITIES.
                                0.100-01
RADIUS TO START SEARCH.
                                0.000 00
RADIUS TO END SEARCH.
                                0.000 00
BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF PIXT ARE
                                                              MULTIPLICITIES
ROOTS OF PIXI
                                                                                           INITIAL APPROXIMATION
  ROOT(1) = 0.99999924980447420 00 + 0.19999984679040900 01 1
                                                                           0.48296291156562790 00 + 0.12940952844381870 00 [
  ROOT( .21 = -0.9921687016776328D 00 + 0.50273964052275640 00 I
                                                                           0.70710675530463460 00 + 0.7071068070684595D 00 I
  RODT(3) = 0.19166896842724220 01 + 0.19669597870445160 01 1
                                                                           0.3882284792654056D 00 + 0.1448888763117193D 01 I
  ROOT ( 4) = 0.2013726806167612D 01 + 0.2086055012105922D 01 1
                                                                           -0.5176382551966724D 00 + 0.19318516083687550 01 f
  ROBT( 5) = 0.20694066410038210 01 + 0.19469500958857110 01 I
                                                                           SOLVED BY DIRECT METHOD
  ROOT( 6) = -0.10154842276975370 01 + 0.49455888811015920 00 I
                                                                           SOLVED BY DIRECT METHOD
AFTER THE ATTEMPT TO IMPROVE ACCURACY. THE ZEROS OF P(K) ARE
                                                              MULTIPL TO ITIES
ROOTS OF P(X)
                                                                                           INITIAL APPROXIMATION
  RODT( 1) = 0.99999962393979710 00 + 0.19999992333489690 01 I
                                                                           0.4829629115656279D 00 + 0.1294095284438187D 00 I
  RDUT ( 2) = -0.9999926539148827D 00 + 0.49999755579552720 00 I
                                                                           0.7071067553046346D 00 + 0.7071068070684595D 00 1
  RDDT( 3) = 0.1999956221578001D 01 + 0.2000020836036100D D1 [
                                                                          . D.3882284792654056D 00 + O.1448888763117193D 01 1
  RDOT ( 4) = 0.2000005829322798D 01 + 0.1999952499501282D 01 I
                                                                           -0.5176382551966724D 00 + 0.1931851608368755D 01 f
  ROOT ( 5) = 0.20000087385097060 01 + 0.19999500538776400 01 1
                                                                           SOLVED BY DIRECT METHOD
 RDOT( 6) = -0.10000053739815870 01 + 0.49999509201216870 00 1
                                                                           SOLVED BY DIRECT METHOD
```

Exhibit 6.5. Roots Are: 2+2i (3), 1+2i (2), -1+.5i (3)

NEWTONS METHOD TO FIND ZERDS OF POLYNOMIALS POLYNOMIAL NUMBER 4 OF DEGREE 12

#### THE CUEFFICIENTS OF PIXE ARE

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.100-03
TEST FOR MULTIPLICITIES. 0.100-01
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF PIX) ARE

ROOTS OF P(X)		MULTIPLICITIES	INITIAL APPROXIMATION	
ROOT( 7) * 0.13937999844005080	01 + -0.54206461303878890 00 I 01 + 0.10233106615266340 01 I 00 + -0.79497020137361440 00 I 00 + -0.13956979981330380 01 I 00 + -0.12069876664549360 01 I	1 0.707106755304 1 0.388228479265 1 -0.517638255196 1 -0.176776714708		00 I 01 I 01 I

IN THE ATTEMPT TO IMPROVE ACCURACY. ROOT( 2) = 0.10200968856097200 01 + ~0.54206461303878890 00 1 DID NOT CONVERGE. THE PRESENT APPROXIMATION AFTER 200 ITERATIONS IS PRINTED BELOW.

IN THE ATTEMPT TO IMPROVE ACCURACY, RODT( 5) = 0.98514220474234140 00 + -0.13956979981330380 01 1 DID NOT CONVERGE. THE PRESENT APPROXIMATION AFTER 200 ITERATIONS IS PRINTED BELOW.

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF PIXE ARE

Exhibit 6.6.

INITIAL APPROXIMATION

```
RODT( 1) = 0.99948113795146260 00 + 0.99950775860617460 00 1 5 0.48296291156562790 00 + 0.1294095284438187D 00 1 RODT( 2) = 0.9952313317176770 00 + -0.10017819876754680 01 1 0.7071067553046346D 00 + 0.7071068070684595D 00 1 0.38822847925540560 00 + 0.148888763117193D 01 1 0.38822847925540560 00 + 0.148888763117193D 01 1 0.38822847925540560 00 + 0.148888763117193D 01 1 0.5176382551966724D 00 + 0.1931851608368755D 01 1 RODT( 5) = 0.9939104218906621D 00 + -0.9941135991702493D 00 1 1 -0.1767767147080701D 01 + 0.1767766758852015D 01 1 RODT( 7) = 0.9960538588565869D 00 + -0.1002759923664572D 01 1 1 -0.289717758307499D 01 + 0.7764567463987070D 00 1 RODT( 7) = 0.1003719838554071D 01 + -0.99631592890632293D 00 1 1 SOLVED 8Y DIRECT METHOD
```

Exhibit 6.6. Roots Are: 1+i(6), 1-i(6)

```
MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL POLYNOMIAL NUMBER 1 OF DEGREE 6
```

#### THE COEFFICIENTS OF PIXE ARE

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.10D-09
TEST FOR MULTIPLICITIES. 0.100-01
RADIUS TO START SEARCH. 0.00D 00
RADIUS TO END SEARCH. 0.00D 00

## BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF PIXE	MULTIPLICITIES	INITIAL APPROXIMATION
RDDT( 11 = 0.1999954888749810D 01 + RODT( 2) = 0.1013502627672869D 01 + RODT( 3) = 0.9866348122507864D 00 + RODT( 4) = -0.1000002106173084D 01 +	0.19961630831608820 01 1 1 1 0.20038068823255210 01 1 1	0.4829629115656279D 00 + 0.1294095284438187D 00 I 0.7071067553046346D 00 + 0.7071068070684599D 00 I SQLVED BY DIRECT METHOD SQLVED BY DIRECT METHOD

AFTER THE ATTEMPT TO IMPROVE ACCURACY

Exhibit 6.7. Roots Are: 2+2i (3), 1+2i (2), -1+.5i

MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL POLYNOMIAL NUMBER 2 OF DEGREE 15

## THE COEFFICIENTS OF PIXI ARE

```
P( 1) = 0.48000000000000000 02 + 0.0000000000000000 00 I P( 2) = 0.25571200000000000 03 + -0.3840000000000000 04 I P( 3) = -0.7353556800000000 03 + -0.3840000000000000 04 I P( 4) = -0.3855565696000000 04 + -0.6946851456000010 04 I P( 5) = -0.17333864648000000 05 + -0.14206259728000000 05 I P( 6) = -0.4947989270400010 05 + -0.1765857464000000 05 I P( 7) = -0.1022394521300000 06 + -0.60306642320000010 04 I P( 8) = -0.1642742200560000 06 + -0.60306642320000010 04 I P( 9) = -0.2036625888420000 06 + -0.10338992276700000 05 I P( 9) = -0.2036625888420000 06 + -0.10338992276700000 06 I P( 11) = -0.127497298590000 06 + -0.1938992276700000 06 I P( 11) = -0.127497298590000 06 + -0.1928689727960000 06 I P( 11) = -0.127434444348000000 05 + -0.1928689727960000 06 I P( 13) = -0.13294344348000000 05 + -0.193802265500000 06 I P( 14) = -0.3539007747000000 05 + -0.193802265500000 06 I P( 15) = -0.18388990200000000 05 + -0.2998889143000000 05 I P( 15) = -0.18388990200000000 00 + -0.29988891430000000 00 I P( 16) = -0.27556200000000000 00 I
```

NUMBER OF INITIAL APPROXIMATIONS GIVEN. D
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.100-01
TEST FOR MULTIPLICITIES. 0.100-01
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

## BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
RODT( 1) = 0.3000000056259372D-02 + 0.6277931932119355D-12 I RODT( 2) = 0.16864729610576890-04 + 0.10000018357040090 01 I RODT( 3) = 0.1280227161896996D-04 + 0.1499987299539190 01 I RODT( 4) = 0.6019678782146761D-04 + 0.2999936661658190D 01 I RODT( 5) = -0.1686685393117943D-04 + 0.9997981635776160D 00 I RODT( 7) = -0.1000163769022359D 01 + -0.1000173771459914D 01 I RODT( 8) = -0.3334905054308976D-03 + -0.150013337339480D 01 I RODT( 10) = 0.3334811914008498D-03 + -0.1999652478539000 I RODT( 10) = 0.3334811914008498D-03 + -0.1499866916248616D 01 I RODT( 11) = -0.1715220973482150D-03 + 0.2999990216209916D 01 I RODT( 11) = -0.1715220973482150D-03 + -0.3000073149878609D 01 I	2 1 1 1 1 2 1 1	0.4829629115656279D 0D + 0.1294095284438187D 0D I 0.7071067553046346D 0D + 0.7071068070684595D 0D I 0.3882284792654056D 0D + 0.1446888763117193D 01 I - 0.5176382551966724D 0D + 0.1767766758852015D 01 I - 0.282977775807499D 01 + 0.7764567483987070D 00 I - 0.3380740248331229D 01 + -0.9058671940816160D 0D I - 0.2828426607107896D 01 + -0.2822842764238439D 01 I - 0.1646848013978999 01 + -0.282842764238439D 01 I 0.1294096345098645D 01 + -0.4829628831453027D 01 I 0.3889088292979509D 01 + -0.388908630D072258D 01 SCM_VED_BY_DIRECT_METHOD
RODT(12) = 0.11131368982465770-03 + 0.30000731498786090 01   RODT(13) = -0.12764927758354440-04 + 0.15000012155052380 01	i	SOLVED BY DIRECT METHOD

AFTER THE ATTEMPT TO IMPROVE ACCURACY

Exhibit 6.8.

ROOTS OF PIXI MULTIPLICITIES INITIAL APPROXIMATION R03T(1) = 0.30000000562581190-02 + 0.82793704892536880-12 ID.4829629115656279D 00 + 0.1294095284438187D 00 1 ROOT! 21 \* 0.3001086172156215D-07 + 0.10000000438390740 01 I 0.7071067553046346D 00 + 0.7071068070684595D 00 I RDDT[3] = 0.61890537903191320-07 + 0.15000001407680000 01 i0.3882284792654056D 00 + 0.1448888763117193D 01 I RODT[ 3] = 0.618995379G319132D-07 + 0.15000001407680000 01 I
RODT[ 4] = 0.2397327410073823D-04 + 0.30000087372252940 01 I
RODT[ 5] = 0.397398454042845D-07 + 0.10000000425568640 01 I
RODT[ 6] = -0.23333333333333340 01 + -0.3027956195429183D-15 I
RODT[ 7] = -0.1000006990404978D 01 + -0.3927956195429183D-15 I
RODT[ 8] = 0.9284422544336306D-08 + -0.1500000131286020 01 I
RODT[ 9] = -0.10000062510065850 01 + -0.10000015374992320 01 I
RODT[ 1] = -0.1571714279305380D-07 + -0.1500000185263290 01 I
RODT[ 1] = -0.2558768623037301D-04 + 0.30000216537917870 01 I
RODT[ 1] = 0.2110926108982832D-04 + 0.30000149073605530 01 I
RODT[ 1] = -0.45537993846927470-07 + 0.149999989437957010 01 I 0.38822544726370380 00 + 0.1498888763117130 01 1
-0.51763825519667240 00 + 0.19318516083687550 01 1
-0.17677671470807010 01 + 0.17677687588520150 01 1
-0.288777775830749900 01 + 0.7764574639870700 00 1
-0.33807402483312290 01 + -0.90586719408161600 00 1 -0.28284266071078960 01 + -0.28284276423843900 01 1 -0.1164684801399899D 01 + -0.434666459873368D 01 [ 0.1294096345098645D 01 + -0.4829628831453027D 01 ] 0.3889088292979509D 01 + -0.3889086300072258D 01 [ SULVED BY DIRECT METHOD RODT(13) - -0.54537993846922740-07 + 0.14999998637957010 01 I SOLVED BY DIRECT METHOD

Exhibit 6.8. Roots Are: -2.33, .003 (2), i(2), 1.5i (2), -1.5i (2) 3i (3), -1-i(3)

```
MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL POLYNOMIAL NUMBER 3 OF DEGREE 8
```

#### THE COEFFICIENTS OF P(X) ARE

```
NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0-100-09
TEST FOR MULTIPLICITIES. 0-100-01
RADIUS TO START SEARCH. 0-000 00
RADIUS TO END SEARCH. 0-000 00
```

## BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APP	ROXIMATION
RDDT( 2) = 0.10005638069089880 01 + 0. RDDT( 3) = -0.10635284187498440 01 + 0. RDDT( 4) = -0.97221225168698249 00 + 0. RDDT( 5) = 0.99943706805064100 00 + 0.	28743998294D 01 I 3 38952568216D 01 I 1 72379199120D 00 I 1 97457973125D 00 I 1 94286928692D 01 I 1 35447909865D 00 I 1	0.4829629115656279D 00 + 0.7071067553046346D 00 + 0.3882284792654056D 00 + -0.5176382551966724D 00 + SOLVED BY DIRECT METHOD SOLVED BY DIRECT METHOD	0.1294095284438187D 00 1 0.70710680706845950 00 1 0.1448888763117193D 01 1 0.19318516083687550 01 1

## AFTER THE ATTEMPT TO IMPROVE ACCURACY

```
ROOTS OF P(X)

ROOTS
```

Exhibit 6.9. Roots Are: 2+2i (3), 1+2i (2), -1+.5i (3)

MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL POLYNOMIAL NUMBER 4 OF DEGREE 12

#### THE COEFFICIENTS OF PIX) ARE

```
NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXINUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.100-01
TEST FOR MULTIPLICITIES. 0.100-01
RABIUS TO START SEARCH. 0.000 00
```

#### BEFORE ATTEMPT TO IMPROVE ACCURACY

```
ROOT: 1 = 0.1004468192948739D 01 + 0.1002186117532751D 01 1 4 0.4829629115656279D 00 + 0.7071068070864595D 00 I 0.7071067553066346D 00 + 0.7071068070864595D 00 I 0.308728473947D DD + 0.98493823789516B 00 I 1 0.3882284792654D56D 00 + 0.1648888763117193D 01 I 0.3882284792654D56D 00 + 0.1448888763117193D 01 I 0.3882284792654D56D 00 + 0.1448888763117193D 01 I 0.3882284792654D56D 00 + 0.1648888763117193D 01 I 0.3882284792654D56D 00 + 0.1648888763117193D 01 I 0.3882284792654D56D 00 + 0.1648888763117193D 01 I 0.70710680706855D 01 I 0.707106807068595D 01 I 0.70710680706855D 01 I 0.7071068070685D 01 I 0.7081085D 01
```

AFTER THE ATTEMPT TO IMPROVE ACCURACY

```
RODTS OF P(X)

RODTS | 1 | = 0.10035767401254570 | 01 + 0.10017426347906060 | 01 | 0.48296291156562790 | 00 + 0.12940952844381870 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950 | 00 | 0.70710680706845950
```

Exhibit 6.10.

```
ROOT( 4) = 0.10007324697602040 01 + -0.99564295721310670 00 I 2 -0.5176382551966724D 00 + 0.19318516083687550 01 I ROOT( 5) = 0.10047654833534890 01 + -0.10006957183589740 01 I 1 -0.176776147080701D 01 + 0.7764567588520150 01 I 0.9994782566274331D 00 + -0.99607660789158700 00 I 1 -0.2897777583074990D 01 + 0.77645674639870700 00 I SOLVED BY DIRECT METHOD
```

Exhibit 6.10. Roots Are: 1+1 (6), 1-i (6)

# GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS

```
NUMBER OF INITIAL APPROXIMATIONS GIVEN. O
MAXIMUM NUMBER OF ITERATIONS.
TEST FOR ZERO IN SUBROUTINE GCD.
                       0.100-02
TEST FOR CONVERGENCE.
                       0-100-09
TEST FOR ZERO IN SUBROUTINE QUAD.
                       0.100-19
TEST FOR MULTIPLICITIES.
                       0.100-01
RADIUS TO START SEARCH.
                       0.00D 00
RADIUS TO END SEARCH.
                       0.000 00
THE DEGREE OF P(X) IS 6 THE COEFFICIENTS ARE
PI5 ) = -0.2800000000000000 02 + 0.58000000000000 02 I
QIX) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE DISTINCT ROOTS OF PIXI.
THE DEGREE OF QIX1 IS 3 THE COEFFICIENTS ARE
Q(3) = -0.2000000000001500 01 + -0.45000000000001180 01 I
Q(2 ) = -0.70000000000004230 01 + 0.3500000000000000000 01 I
0(1 ) = 0.999999999997762D 00 + 0.699999999999812D 01 I
ROOTS OF Q(X)
                                                                 INITIAL APPROXIMATION
RODT ( 1) = 0.9999999999975650 00 + 0.19999999999574D 01 I
                                                       0.4829629115656279D 00 + 0.1294095284438187D 00 I
RD(T(2)) = 0.20000000000003470 01 + 0.20000000000005290 01 T
                                                        RESULTS OF SUBROUTINE QUAD
RESULTS OF SUBROUTINE QUAD
                                            MULTIPLICITIES
ROOTS OF P(X)
                                                                 INITIAL APPROXIMATION
RDDT( 1) = 0.999999999997565D 00 + 0.199999999999574D 01 I
                                                      0.48296291156562790 00 + 0.1294095284438187D 00 I
                                                2
RODT(2) = 0.200000000000003470 01 + 0.2000000000005300 01 I
                                                3
                                                       RESULTS OF SUBROUTINE QUAD
RESULTS OF SURROUTINE QUAD
```

# GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS POLYNOMIAL NUMBER $\ 2$

```
NUMBER OF INITIAL APPROXIMATIONS GIVEN.
MAXIMUM NUMBER OF ITERATIONS.
TEST FOR ZERO IN SUBROUTINE GCD.
                                     0.100-02
TEST FOR CONVERGENCE.
                                      0-100-09
TEST FOR ZERO IN SUBROUTINE QUAD. 0.100-19
TEST FOR MULTIPLICITIES.
                                     0.100-01
RADIUS TO START SEARCH.
                                     0-000-00
RADIUS TO END SEARCH.
                                     0.000 00
THE DEGREE OF PIXT IS 15 THE COEFFICIENTS ARE
 P(15) = 0.25571200000000000 03 + -0.3840000000000000 03 1

P(14) = -0.7353556800000000 02 + -0.218969600000000 04 1

P(13) = -0.3855565690000000 04 + -0.6946851456000010 04 1

P(12) = -0.17333866648000000 05 + -0.14206259728000000 05 1
 P(11) = -0.49679892704000010 05 + -0.17658574640000000 05 I
 P(10) = -0.1022394522130000D 06 + -0.6030664232000001D 04 T
 P(9 ) = -0.16427422095600000 06 + 0.41373662304000000 05 J
P(8 ) = -0.20366258884200000 06 + 0.10938992276700000 06 J
P(7) = -0.18712557800100000 06 + 0.19298654403300000 06 1
P(6) = -0.12749972985900000 06 + 0.21713412274200000 06 1
P(5) = -0.28146927168000000 05 + 0.19284897279600000 06 1
 P(4) = 0.13294344348000000 05 + 0.10381302265500000 06 1
 P(3 ) = 0.30539007747000000 05 + 0.29989891413000000 05 I
 P(2) = -0.18358990200000000 03 + -0.18276321600000000 03 I
 DIXI IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE DISTINCT ROOTS OF PIXI.
THE DEGREE OF Q(X) IS 7 THE COEFFICIENTS ARE
 0(7) = 0.1598559999590830 03 + -0.14400000000184630 03 1
 Q(6 ) = 0.2675199999866287D 03 + -0.5275680000068238D 03 1
 Q(5) = 0.32719599997819040 03 + -0.914416000000759420 03 J
 Q(4 ) = 0.23015999960445530 02 + -0.1521252000003768D 04 I
 Q13 1 = -0.7207200004799688D 02 + -0.132742799999659D 04 1
 Q(2 ) = -0.7557840000541563D 03 + -0.75200400003332210 03 [
 Q(1 ) = 0.22680001335502310 01 + 0.2267999925828633D 01 1
ROOTS OF DIXI
                                                                                                         INITIAL APPROXIMATION
 RDGT( 1) = 0.30000000386459970-02 + -0.1370543176638413D-09 !
                                                                                       0.4829629115656279D 00 + 0.1294095284438187D 00 1
RODT( 4) = 0.1167767202194363D-10 + -0.150000000000067630 01 [
                                                                                       0.7071067553046346D 00 + 0.7071068070684595D 00 1
                                                                                       0.3882284792654056D 00 + 0.1448888763117193D 01 1
                                                                                      -0.5176382551966724D 00 + 0:1931851608368755D 01 [
```

Exhibit 6.12.

```
ROOT( 5) = -0.2333333333340885D 01 * 0.4292301395925554D-11 1 ROOT( 6) = 0.1140452177139650D-09 * 0.3000000000053944D 01 I RESULTS OF SUBROUTINE QUAD

ROOTS OF PIX!

MULTIPLICITIES

INITIAL APPROXIMATION

ROOT( 1) = 0.3000000038645997D-02 * -0.1370543176638413D-09 I 2 0.4829629115656279D 00 * 0.129409528443B1B7D 00 I ROOT( 2) = 0.1466637065231162D-09 * 0.100000000302556D 01 I 2 0.7071067553046346D 00 * 0.7071068070684595D 00 I ROOT( 3) = -0.1626208545466292D-09 * 0.149999999807066D 01 I 2 0.3882284792654056D 00 * 0.1448888763117193D 01 I ROOT( 4) = -0.11677692475724520-10 * -0.1500000000006763D 01 I 2 0.388228479265056D 00 * 0.1448888763117193D 01 I ROOT( 5) = -0.2333333333340885D 01 * 0.429155150097099D-11 I 1 0.51767807410D 01 * 0.1767766758852015D 01 I ROOT( 6) = 0.114045567560921DD-09 * 0.300000000053944D 01 I 3 RESULTS OF SUBROUTINE QUAD

Exhibit 6.12. Roots Are: -2.33, .003 (2), i(2), 1.5i (2), -1.5i (2) 3i (3), -1-i (3)
```

# GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS POLYNOMIAL NUMBER 3

```
NUMBER OF INITIAL APPROXIMATIONS GIVEN.
MAXIMUM NUMBER OF ITERATIONS.
TEST FOR ZERO IN SUBROUTINE GCD.
                                 0.100-02
                                 0.100-09
TEST FOR CONVERGENCE.
TEST FOR ZERO IN SUBROUTINE QUAD. 0.100-19
TEST FOR MULTIPLICITIES.
                                 0-100-01
RADIUS TO START SEARCH.
                                 0.000 00
                                 0.000 00
RADIUS TO END SEARCH.
THE DEGREE OF PIX) IS B THE COEFFICIENTS ARE
 P(7) = -0.51750000000000000 02 + 0.430000000000000 02 I

P(6) = 0.15725000000000000 03 + 0.14462500000000000 03 I

P(5) = 0.3075000000000000 03 + -0.3475000000000000 03 I
 P(3 1 = -0.5857500D0000000010 03 + 0.42475000D000000010 03 I
P(2 ) = 0.181000000000000000 03 + 0.4420000000000010 03 I
 Q(X) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE DISTINCT ROOTS OF PIXI.
THE DEGREE OF Q(X) IS 3 THE COEFFICIENTS ARE
 Q(3) = -0.2000000000000002730 01 + -0.45000000000002750 01 1
 Q12 ) = -0.70000000000009350 Q1 + 0.35000000000004690 Q1 I
 Q11 J = 0.99999999999941730 G0 + 0.70000000000004980 G1 I
                                                                                            INITIAL APPROXIMATION
ROOTS OF Q(X)
  RODT ( 1) ≈ 0.999999999995483D OD + 0.19999999999755D O1 [
                                                                             0.48296291156562790 00 + 0.12940952844381870 00 1
                                                                               RESULTS OF SUBROUTINE QUAD
RESULTS OF SUBROUTINE QUAD
 RDOT( 2) = 0.2000D000000000650 01 + 0.2000D0D000005390 01 1
 ROOT ( 3) = -0.9999999999994080 00 + 0.499999999998140 00 I
                                                              MULTIPLICITIES
                                                                                           INITIAL APPROXIMATION
ROOTS OF PIX)
                                                                            0.4629629115656279D 00 + 0.1294095284438187D 0D 1
RESULTS OF SUBROUTINE QUAD
 RDOT( 1) = 0.9999999999995483D 00 + 0.199999999999755D 01 I
 RODT ( 2) = 0.2000000000000650 01 + 0.2000000000005390 01 I
 ROOT( 31 = -0.99999999999994080 00 + 0.499999999998120 00 I
                                                                              RESULTS OF SUBROUTINE QUAD
```

Exhibit 6.13. Roots Are: 2+2i (3), 1+2i (2), -1+.5i (3)

# GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS POLYNOMIAL NUMBER 4

```
NUMBER OF INITIAL APPROXIMATIONS GIVEN.
MAXIMUM NUMBER OF ITERATIONS.
TEST FOR ZERO IN SUBROUTINE GCO.
                            0.100-02
                            0.100-09
TEST FOR CONVERGENCE.
TEST FOR ZERO IN SUBROUTINE QUAD.
                            0.100~19
                            0.100-01
TEST FOR MULTIPLICITIES.
                            0.000 00
RADIUS TO START SEARCH.
RADIUS TO END SEARCH.
                            0.00D D0
THE DEGREE OF PIX) IS 12 THE COEFFICIENTS ARE
 P(12) = 0.12000000000000000 02 + 0.00000000000000 00 I

P(11) = 0.7200000000000000 02 + 0.000000000000000 00 I

P(10) = 0.280000000000000 03 + 0.00000000000000 00 I

P(9) = 0.7800000000000000 03 + 0.00000000000000 00 I
 P(7 ) = 0.262400000000000 04 + 0.000000000000000 00 I

P(6 ) = -0.326400000000000 04 + -0.000000000000000 00 I

P(5 ) = 0.312000000000000 04 + 0.000000000000000 00 I
 P(3 ) = 0.11520000000000000 04 + 0.0000000000000000 00 I
 DIX) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE DISTINCT ROOTS OF PIXI.
THE DEGREE OF Q(X) IS 2 THE COEFFICIENTS ARE
 0(2) 1 = 0.1999999999998300 01 + 0.0000000000000000 00 I
                                                     MULTIPLICITIES
RBOYS OF PIXE
                                                                   RESULTS OF SUBROUTINE QUAD
 RESULTS OF SUBROUTINE QUAD
```

Exhibit 6.14. Roots Are: 1+i (6), 1-i (6)

# GREATEST COMMON DIVISOR METHOD TUSED WITH MOLVERS METHOD TO FIND ZEROS OF POLYNOMIALS POLYNOMIAL NUMBER 1

```
NUMBER OF INITIAL APPROXIMATIONS GIVEN.
MAXIMUM NUMBER OF ITERATIONS.
TEST FOR ZERO IN SUBROUTINE GCD.
                             0-100-02
TEST FOR CONVERGENCE.
                             0-100-09
TEST FOR ZERO IN SUBROUTINE QUAD.
                             0.100-19
TEST FOR MULTIPLICITIES.
                             0.100-01
                             0.000 00
RADIUS TO START SEARCH.
RADIUS TO END SEARCH.
                             0.000 00
THE DEGREE OF PIXI-IS 6 THE COEFFICIENTS ARE
P(5 ) = -0.28000000000000000 02 + 0.58000000000000 02 1
P(4 ) = 0.17100000000000000 03 + 0.1500000000000000 01 1
 GIX) IS THE POLYMONIAL WHICH HAS AS ITS ROOTS THE DISTINCT ROOTS OF PIX).
THE DEGREE OF DIX) IS 3 THE COEFFICIENTS ARE
 Q(2) = -0.70000000000004230 01 + 0.350000000000000000 01 I
Q(1) = 0.999999999977620 00 + 0.6999999999999120 01 I
                                                                                 INITIAL APPROXIMATION
ROOTS OF GIX!
 ROOT( 1) = 0.99999999999975650 00 + 0.199999999999740 01 (
                                                                     0.4829629115656279D 00 + 0.1294095284438187D 00 1
 ROOT( 21 = 0.200000000000003470 01 + 0.20000000000005290 01 I
                                                                     SOLVED BY DIRECT METHOD
 RDOT( 3) = -0.99999999999999539D 00 + 0.500000000000149D 00 I
                                                                     SOLVED BY DIRECT METHOD
                                                       MULTIPLICITIES
                                                                                 INITIAL APPROXIMATION
ROOTS OF P(X)
                                                                     0.4829629115656279D 00 + 0.1294095284438187D 00 I
 ROOT( 1) = 0.9999999999997568D CO + 0.199999999999574D OI 1
                                                            2
 ROOT( 2) = 0.200000000000003470 01 + 0.2000000000005300 01 1
                                                                     RESULTS OF SUBROUTINE QUAD RESULTS OF SUBROUTINE QUAD
                                                            3
 ROUT( 3) = -0.99999999999999410 00 + 0.5000000000001480 00 [
```

Exhibit 6.15. Roots Are: 2+2i (3), 1+2i (2), -1+.5i

#### GREATEST COMMON DIVISOR METHOD USED WITH MULLERS METHOD TO FIND ZEROS OF POLYNOMIALS POLYNOMIAL NUMBER 2

```
NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
HAXIMUM NUMBER OF ITERATIONS.
                                        200
. TEST FOR ZERO IN SUBROUTINE GCO.
                                   0.10D-02
TEST FOR CONVERGENCE.
                                   0.100-09
TEST FOR ZERO IN SUBROUTINE QUAD.
TEST FOR MULTIPLICITIES.
                                   0.10D-01
RADIUS TO START SEARCH.
                                   0.000 00
RADIUS TO END SEARCH.
                                   0.00D 00
THE DEGREE OF P(X) IS 15 THE COEFFICIENTS ARE
```

```
P(15) = 0.2557[20000000000 03 + -0.3840000000000000 03 1
P(14) = -0.73535568000000000 02 + -0.21896960000000000 04 1
P(13) = -0.38555656960000000 04 + -0.69468514560000010 04 i
P(12) = -0.17333864646000000 05 + -0.14206259728000000 05 T
P(11) = -0.1973894504000010 05 + -0.17658574640000000 05 I
P(10) = -0.10223945221300000 06 + -0.60306642320000010 06 I
P(9) = -0.16427422005600000 06 + 0.41373662304000000 05 I
P(8) = -0.20366258884200000 06 + 0.10938992276700000 06 I

P(7) = -0.18712557800100000 06 + 0.19298654403300000 06 I
P(6 ) = -0.12749972985900000 06 + 0.21713412274200000 06 I
P(5 ) = -0.28146927168000000 05 + 0.19284897279600000 06 I
P(4 ) = 0.13294344348000000 05 + 0.10381302265500000 06 1
P(3 ) = 0.30539007747000000 05 + 0.29989891413000000 05 1
P(2 ) = -0.18358990200000000 03 + -0.18276321600000000 03 1
P(1 ) = 0.27556200000000000 00 + 0.27556200000000000 00 J
```

## Q(x) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE DISTINCT ROOTS OF P(X). THE DEGREE OF Q(X) 15 7 THE COEFFICIENTS ARE

```
Q(7 ) = 0.1598559999959083D 03 + -0.14400000000018463D 03 1
Q(6 ) = 0.2675199999866287D 03 + -0.5275680000068238D 03 I
Q(5 ) = 0.32719599997819040 03 + -0.91441600000759420 03 I
Q(4 ) = 0.23015999960445530 02 + -0.15212520000037680 04 I
0(3 ) = -0.72072000047996880 02 + -0.1327427999996590 04 I
0(2 ) = -0.75578400005415630 03 + -0.75200400003332210 03 J
011 1 = 0.2268000133550231D 01 + 0.22679999258286330 01 [
```

## ROOTS OF Q(X)

```
RODT( 1) = 0.30000000386459970-02 + -0.13705431778101480-09 1
ROOT( 2) = 0.14666372491173318-09 + 0.10000000003025560 01 I
ROOT( 3) = -0.1626210146532216D-09 + 0.1499999998070670 01 I
ROOT( 4) = 0.1140454591987512D-09 + 0.30000000000539440 01 I
```

#### INITIAL APPROXIMATION

```
0.48296291156562790 00 + 0.12940952844381870 00 [
0.70710675530463460 00 + 0.7071068070684595D 00 1
0.3882284792654056D DD + 0.1448888763117193D D1 1
-0.5176382551966724D 00 + 0.1931851608368755D 01 f
```

```
RODT( 5) = -0.23333333333408840 01 + 0.42926697510937080-11 f RODT( 6) = -0.1167725175813909D-10 + -0.15000000000067630 01 I ROOT( 7) = -0.10000000000322610 01 + -0.9999999998557870 00 I
                                                                                                                                                                                                                                             SOLVED BY DIRECT METHOD
ROOTS OF PIXE
                                                                                                                                                                                             MULTIPLICITIES
                                                                                                                                                                                                                                                                                       INITIAL APPROXIMATION
 RODY( 1) = 0.3000000038645997D-02 + -0.13705431778187450-09 I
RODY( 2) = 0.14666360907717090-09 + 0.10000000003025560 01 I
RODY( 3) = -0.16262091348697400-09 + 0.14999999998070660 01 I
RODY( 4) = 0.1140455658084308D-09 + 0.30000000000539440 01 I
RODY( 6) = -0.23333333333468850 01 + 0.42914027148801489-11 I
RODY( 6) = -0.11677735717066360-10 + -0.150000000000067630 D1 I
RODY( 7) = -0.100000000000322600 01 + -0.99999999998557790 00 I
                                                                                                                                                                                                                                          0.48296291156562790 00 + 0.12940952844381870 00 I

0.70710675530463460 00 + 0.70710680706845950 00 I

0.38822847926540560 00 + 0.14938887631171930 01 I

-0.51763825519667240 00 + 0.19318516083687550 01 I

-0.17677671470807010 01 + 0.17677667588520150 01 I
                                                                                                                                                                                                                                            RESULTS OF SUBROUTINE QUAD
RESULTS OF SUBROUTINE QUAD
                                               Exhibit 6.16. Roots Are: -2.33, .003 (2), i (2), 1.5i (2), -1.5i (2) 3i (3), -1-i (3)
```

-0.1767767147080701D 01 + 0.1767766758852015D 01 1

SOLVED BY DIRECT METHOD

# GREATEST COMMON DIVISOR METHOD USED WITH MULLERS METHOD TO FIND ZEROS OF POLYNOMIALS POLYNOMIAL NUMBER 3

NUMBER OF INITIAL APPROXIMATIONS GIVEN.

```
MAXIMUM NUMBER OF ITERATIONS.
TEST FOR ZERO IN SUBROUTINE GCD.
                              0.100-02
TEST FOR CONVERGENCE.
                              0-100-09
TEST FOR ZERO IN SUBROUTINE QUAD. 0.100-19
TEST FOR MULTIPLICITIES.
                              0-100-01
                              0-000-00
RADIUS TO START SEARCH.
                              0.000 00
RADIUS TO END SEARCH.
THE DEGREE OF P(X) IS 8 THE COEFFICIENTS ARE
 P(3) = -0.58575000000000010 03 + 0.42475000000000010 03 I
P(2) = 0.181000000000000000 03 + 0.4420000000000010 03 I
 QIX) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE DISTINCT ROOTS OF P(X). THE DEGREE OF Q(X) IS 3 THE COEFFICIENTS ARE
 013 + -0.200000000000002730 01 + -0.45000000000002750 01 1
 Q(2) = -0.7000000000000009350 01 + 0.35000000000004690 01 1
 Q(1 1 = 0.9999999999941730 00 + 0.7000000000004980 01 I
                                                                                  INITIAL APPROXIMATION
ROOTS OF Q(X)
                                                                      0.48296291156562790 00 + 0.12940952844381870 00 [
SDLVED BY DIRECT METHOD
 ROOT( 3) = -0.9999999999999908D 00 + 0.4999999999999814D 00 1
                                                                      SOLVED BY. DIRECT METHOD
                                                        MULTIPLICITIES
                                                                                  INITIAL APPROXIMATION
ROOTS, OF PEXA
                                                                      0.4829629115656279D 00 + 0.1294095284438187D 00 I
 ROOT( 1) = 0.9999999999995483D 00 + 0.1999999999999755D 01 1
                                                                      RESULTS OF SUBROUTINE QUAD
 RBUT(2) = 0.200000000000000665D 01 + 0.200000000000539D 01 I
 ROOT ( 3) = -0.9999999999994100 00 + 0.49999999999998120 00 [
                                                                      RESULTS OF SUBROUTINE QUAD
```

Exhibit 6.17. Roots Are: 2+2i (3), 1+2i (2), -1+.5i (3)

# GREATEST COMMON-01415OR METHOD USED-WITH MULLERS METHOD TO FIND ZERDS OF POLYNOMIALS POLYNOMIAL NUMBER 4

```
NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR ZERO IN SUBROUTINE GCD. 0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD. 0.10D-19
TEST FOR MULTIPLICITIES. 0.10D-01
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00
```

#### THE DEGRÉE OF PIX) IS 12' THE COEFFICIENTS ARE

DIX) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE DISTINCT ROOTS OF P(X). THE DEGREE OF Q(X) IS 2 THE COEFFICIENTS ARE

ROOTS OF P(X)

#### MULTIPLICITIES

```
RODT | 1 | = 0.1000000000000000000 01 + 0.9999999999999740 00 | 6 RESULTS OF SUBROUTINE QUAD ROOT | 2 | = 0.100000000000000 01 + -0.99999999999990740 00 | 6 RESULTS OF SUBROUTINE QUAD
```

Exhibit 6.18. Roots Are: 1+1(6), 1-1 (6)

```
NUMBER OF INITIAL APPROXIMATIONS GIVEN. O
 MAXIMUM NUMBER OF ITERATIONS.
                            200
 TEST FOR ZERO IN SUBROUTINE GCD.
                        0.100-02
 TEST FOR CONVERGENCE.
                         0.10D-09
 TEST FOR ZERO IN SUBROUTINE QUAD.
                        0.100-19
 RADIUS TO START SEARCH.
                         0.000 00
 RADIUS TO END SEARCH.
                         0.00D 00
 THE DEGREE OF PLX) IS 6 THE COEFFICIENTS ARE
  THE FOLLOWING POLYNOMIAL. G(X). CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY 1
  GII 1 = 0.9999999999973500 00 + -0.50000000000175420 00 I
 RODTS OF PIXI
                                             MULTIPLICITIES
                                                                 INITIAL APPROXIMATION
  RDDT( 1) = -0.999999999999973500 00 + 0.50000000000175420 00 I
                                                             NO INITIAL APPROXIMATIONS
 THE FOLLOWING POLYNOMIAL, GIXI, CONTAINS ALL THE ROOTS OF PIXI WHICH HAVE MULTIPLICATY 2
  6(2 ) = 0.1000000000000000 D1 + 0.000000000000000 00 t
  G(1) = -0.9999999999991790 00 + -0.1999999999968450 01 1
```

REPEATED USE OF THE GREATEST COMMON DIVISOR AND NEWTONS METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYMONIALS

POLYNOMIAL NUMBER 1

Exhibit 6.19.

Exhibit 6.19. Roots Are: 2+2i (3), 1+2i (2), -1+.5i

# REPEATED USE OF THE GREATEST COMMON DIVISOR AND NEWTONS METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIALS POLYNOMIAL NUMBER 2

```
NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR ZERO IN SUBROUTINE GCD. 0.100-02
TEST FOR CONVERGENCE. 0.100-09
TEST FOR ZERO IN SUBROUTINE QUAD. 0.100-19
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00
```

#### THE DEGREE OF PEXTIS 15 THE COEFFICIENTS ARE

THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY

G(2 ) \* 0.48000000000000000 D2 + 0.00000000000000000 D0 I G(1 ) = 0.11199999928960310 D3 + -0.82749886587407680-06 I

MULTIPLICITIES

'INITIAL APPROXIMATION

RODT ( II = -0.23333333185333970 OI + 0.17239559705709930-07 I

NO INITIAL APPROXIMATIONS

Exhibit 6.20.

```
THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY 2
G(4) = -0.29999706558634380-02 + -0.99999996563627250 00 1
G(3 ) = 0.2249999931178885D 01 + 0.3000058598117361D-02 [
G12 ) = -0.67501099024602780-02 + -0.22500001290676650 01 I
G(1 ) = 0.4643479465382683D-06 + 0.6749603982170172D-02 I
ROOTS OF GIXT .
                                                                                  INITIAL APPROXIMATION
                                                                    0.48296291156562790 00 + 0.1294095284438187D 00 I
0.7071067553046346D 00 + 0.70710680706845950 00 I
RESULTS OF SUBROUTINE QUAD
 ROOT( 1) = 0.2999823203921417D-02 + -0.2056988264688649D-06 [
ROST( 2) = 0.47675818456966510-06 + 0.10000005558501840 01 i
ROST( 3) = 0.23317216169336970-07 + -0.14999999724317250 01 [
 ROOT ( 4) = -0.35262345871804790-06 + 0.14999995879166400 01 1
                                                                        RESULTS OF SUBROUTINE QUAD
ROOTS OF P(X)
                                                       MULTIPLICITIES
                                                                                 INITIAL APPROXIMATION
                                                                   0.4829629115656279D 00 + 0.1294095284438187D 00 1 0.7071067553046346D 00 + 0.7071068070684595D 00 X
 ROOT(1) = 0.2999823203921417D-02 + -0.2056988264688661D-06 
 ROST ( 2) * 0.4767581845685002D-06 + 0.1000000555850184D 01 [
 ROOT ( 3) * 0.23409171910427910-07 + -0.1499999972431725D 01 T
                                                                            NO INITIAL APPROXIMATIONS
NO INITIAL APPROXIMATIONS
                                                            2
 RODT( 4) = -0.3527154144585488D-06 + 0.1499999587916640D 01 1
THE FOLLOWING POLYNOMIAL, GIX). CONTAINS ALL THE ROOTS OF PIX) WHICH HAVE MULTIPLICITY 3
G(2) = 0.9999998537055410 00 + -0.20000000171626320 01 I
G(1) = 0.300000000025333350 01 + -0.30000000317915140 01 1
ROOTS OF PIXE
                                                       MULTIPLICITIES
                                                                                 INITIAL APPROXIMATION
RDDT( 1) = 0.1462930432349907D-07 + 0.30000000171626330 01 I
                                                                            NO INITIAL APPROXIMATIONS
NO INITIAL APPROXIMATIONS
             Exhibit 6.20 Roots Are: -2.33, .003 (2), i (2), 1.5i (2),
                                 -1.5i (2), 3i (3), -1-i (3)
```

```
REPEATED USE OF THE GREATEST COMMON DIVISOR AND NEWTONS METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIALS
            POLYNOMIAL NUMBER 3
NUMBER OF INITIAL APPROXIMATIONS GIVEN. O
MAXIMUM NUMBER OF ITERATIONS.
TEST FOR ZEPO IN SUBROUTINE GCD.
                                         0.10D-02
TEST FOR CONVERGENCE.
                                          0.100-09
TEST FOR ZERO IN SUBROUTINE QUAD. 0.100-19
RADIUS TO START SEARCH.
                                          0.000 00
RADIUS TO END SEARCH.
                                          0.00D 00
THE DEGREE OF P(X) IS 8 THE COEFFICIENTS ARE
 P(8 ) = -0.5000000000000000000 01 + -0.1150000000000000 02 I

P(7 ) = -0.51750000000000000 03 + 0.14462500000000000 03 I

P(6 ) = 0.15725000000000000 03 + -0.347500000000000 03 I

P(5 ) = 0.3075000000000000 03 + -0.4948750000000000 03 I

P(4 ) = -0.4952500000000000 03 + -0.4948750000000000 03 I

P(3 ) = -0.58575000000000000 03 + 0.4247500000000001 03 I

P(2 ) = 0.181000000000000000 03 + 0.442000000000001 03 I
 NO ROOTS OF MULTIPLICITY 1
THE FOLLOWING POLYNOMIAL. G(X). CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY 2
```

Exhibit 6.21.

MULTIPLICITIES

INITIAL APPROXIMATION

NO INITIAL APPROXIMATIONS

ROOT( 1) = 0.9999999999464980 00 + 0.199999999946880 01 I

ROOTS OF PIXE

REPEATED USE OF THE GREATEST COMMON DIVISOR AND NEWTONS. METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIALS POLYNOMIAL NUMBER: 4

```
NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR ZERO IN SUBROUTINE GCD. 0.100-02
TEST FOR ZERO IN SUBROUTINE QUAD. 0.100-19
TRADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00
```

#### THE DEGREE OF P(X) IS 12 THE COEFFICIENTS ARE

NO ROOTS OF MULTIPLICITY 1

NO ROOTS OF MULTIPLICITY 2

Exhibit 6.22.

```
NO ROOTS OF MULTIPLICITY 3
NO ROOTS OF MULTIPLECITY 4
NO ROOTS OF MULTIPLICITY 5
THE FOLLOWING POLYNOMIAL, GIXI, CONTAINS ALL THE ROOTS OF PIXI WHICH HAVE MULTIPLICITY 6
MULTIPLICITIES
                                                                      INITIAL APPROXIMATION
ROOTS OF P(X)
ROOT( 1) = 0.1000000000000033D 01 + 0.999999999999706D 00 I
ROOT( 2) + 0.1000000000000033D 01 + -0.99999999999706D 00 I
                                                                  NO INITIAL APPROXIMATIONS
                                                                  NO INITIAL APPROXIMATIONS
```

Exhibit 6.22. Roots Are: 1+i (6), 1-i (6)

```
REPEATED USE OF THE GREATEST COMMON DIVISOR AND MULLERS. METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIALS
       POLYNOMIAL NUMBER I
MAXIMUM NUMBER OF ITERATIONS.
TEST FOR ZERO IN SUBROUTINE GCD.
                         0.100-02
TEST FOR CONVERGENCE.
                        0.100-09
TEST FOR ZERO IN SUBROUTINE QUAD. 0.100-19
RADIUS TO START SEARCH.
                         0.000 00
RADIUS TO END SEARCH.
                         0.000 00
THE DEGREE OF PIXI-IS & THE COEFFICIENTS ARE
P(1) = -0.7300000000000000 03 1
P(2) = -0.2280000000000000 03 + 0.10400000000000 03 1
P(1) = 0.7200000000000000 02 + 0.104000000000000 03 1
THE FOLLOWING POLYNOMIAL, GIX). CONTAINS ALL THE ROOTS OF PIX1 WHICH HAVE MULTIPLICITY 1
G(1 ) = 0.999999999999973500 '00 + -0.5000000000175420 00 1
                                                                  INITIAL APPROXIMATION
ROOTS OF GIXT
0.48296291156562790 00 + 0.12940952844381870 00 1
                                              MULTIPLICITIES
ROOTS OF PIXT
                                                                   . INITIAL APPROXIMATION
```

0.4829629115656279D DO + 0.1294095284438187D OO 1

Exhibit 6.23.

ROOT( 1) = -0.9999999999999999973500 00 + 0.50000000000175410 00 1

```
THE FOLLOWING POLYNOMIAL, GIX), CONTAINS ALL THE ROOTS OF PIX) WHICH HAVE MULTIPLICITY 2
G(1) = -0.99999999999991790 00 + -0.1999999999998450 01 T
                                                                  INITIAL APPROXIMATION
ROOTS OF GIXI
                                                        0.48296291156562790 00 + 0.12940952844381870 00 I
INITIAL APPROXIMATION
                                            MULTIPLICITIES
ROOTS OF P(X)
ROUT( 1) = 0.999999999991790 00 + 0.1999999999968450 01 I
                                                        0.4829629115656279D 00 + 0.1294095284438187D 00 I
****************************
THE FOLLOWING POLYNOMIAL. GIXI. CONTAINS ALL THE ROOTS OF PIXI WHICH HAVE MULTIPLICITY 3
1 ) = -0.199999999999967D OL + -0.200000000001519D OL I
                                            MULTIPLICITIES
                                                                INITIAL APPROXIMATION
ROOTS OF PIXE
RDOT( 3) = 0.1999999999999967D 01 + 0.200000000001519D 01 1
                                                        NO INITIAL APPROXIMATIONS
               Exhibit 6.23. Roots Are: 2+2i (3), 1+2i (2), -1+.5i
```

REPEATED USE OF THE GREATEST COMMON DIVISOR AND MULLERS METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIALS

```
NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0

MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR ZERO IN SUBROUTINE GCD. 0.10D-02
TEST FOR CONVERGENCE. 0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD. 0.10D-19
RADIUS TO START SEARCH. 0.00D 00

THE DEGREE OF P(X) IS 15 THE COEFFICIENTS ARE
```

```
        P(16) =
        0.480000000000000000
        02 +
        0.0000000000000000
        00 I

        P(15) =
        0.25571200000000000
        03 +
        -0.384000000000000
        03 I

        P(14) =
        -0.7353556800000000
        04 +
        -0.6946851456000000
        04 I

        P(13) =
        -0.3855565696000000
        05 +
        -0.14266259728000000
        04 I

        P(11) =
        -0.496799270400000
        05 +
        -0.14266259728000000
        05 I

        P(11) =
        -0.496799270400000
        06 +
        -0.6030664232000000
        06 I

        P(10) =
        -0.1022394522130000
        06 +
        -0.60306642320000000
        06 I

        P(8) |
        -0.293625388420000
        06 +
        0.41373662304000000
        05 I

        P(4) |
        -0.236625388420000
        06 +
        0.1929865440330000
        06 I

        P(4) |
        -0.12749972985900000
        06 +
        0.19298654403300000
        06 I

        P(5) |
        -0.2864997276985900000
        06 +
        0.19298654403300000
        06 I

        P(4) |
        -0.13294344348000000
        05 +
        0.19284897279600000
        06 I

        P(4) |
        -0.132943443438000000
        05 +
        <
```

THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY |

ROOTS OF G(X)

INITIAL APPROXIMATION

ROOT( 1) = -0.23333333185333970 DI + 0.17239559705709930-07 I

0.4829629115656279D 00 + 0.1294095284438187D 00 I

ROOTS, OF PEXT

MULTIPLICITIES

INITIAL APPROXIMATION

Exhibit 6.24.

```
RDDT( 1) = -0.2333333318533397D 01 + 0.1723955970570993D-07 I
                                                                               0.4829629115656279D 00 + D.1294095284438187D 00 1
THE FOLLOWING POLYNOMIAL. G(K). CONTAINS ALL THE ROOTS OF PIX) WHICH HAVE MULTIPLICITY 2
G15 } = 0.1000BD000BD00BB 01 + 0.000BD00BBD0B D0 E
G14 1 = -0.2999706558634380-02 + -0.999999656362725D 0D E
G(3 } = 0.2249999931178885D 01 + 0.300058598117361D-02 E
 G(Z ) = -0.67501099024602780-02 + -0.22500001290676650 01 1
 Gt1 1 = 0.46434794653826830-06 + 0.67496039821701720-02 1
                                                                                             INITIAL APPROXIMATION
ROOTS OF G(X)
 ROOT( 1) = 0.29998232039214170-02 + -0.20569882646886600-06 [
                                                                               0.4829629115656279D 00 + 0.12940952844381870 00 1
                                                                               0.70710675530463460 00 + 0.7071068070684595D 00 I
 RDDT(2) = 0.47675818456954060-06 + 0.100000005558501840 01 I
                                                                               SOLVED BY DIRECT METHOD
 ROOT ( 3) = 0.23317216169372120-07 + -0.14999999724317250 D1 I
ROOT( 4) = -0.35262345871801270-06 + 0.14999995879166400 01 1
                                                                                SOLVED BY DIRECT METHOD
                                                               MULTIPLICITIES
                                                                                             INITIAL APPROXIMATION
ROOTS OF PIXT
 RDOT( 1) = 0.29998232039214170-02 + -0.2056988264688650D-06 I
                                                                                0.4829629115656279D 00 + 0.129409528443B187D D0 I
                                                                               0.70710675530463460 00 + 0.70710680706845950 00 1
 RODT( 2) = 0.47675818456831650-06 + 0.10000005558501840 01 I
                                                                                NO INITIAL APPROXIMATIONS
 ROOT(3) = 0.23409171910410740-07 + -0.14999999724317250 01 1
                                                                               NO INITIAL APPROXIMATIONS
 ROOT( 4) = -0.35271541445867050-06 + 0.14999995879166400 01 1
THE FOLLOWING POLYNOMIAL, GIX), CONTAINS ALL THE ROOTS OF PIX! WHICH HAVE MULTIPLICITY 3
G(2 ) = 0.9999998537055410 00 + -0.20000000171026320 01 [
G(1 ) = 0.30000000025333350 01 + -0.30000000317915140 01 [
                                                                                             INITIAL APPROXIMATION .
```

Exhibit 6.24.

ROOTS OF GIXT

-8-080448296294355562790-00-4 00-12940952844381870 00 I SOLVEO BY DIRECT METHOD

ROOTS OF PEXT

MULTIPLICITIES

INITIAL APPROXIMATION

ROOT( 1) = -0.999999999998586D 00 + -0.10000000000000000 01 1 ROOT( 2) = 0.1462930456963442D-07 + 0.3000000017162634D 01 [ 0.4829629115656279D 00 + 0.1294095284436187D 00 I ND INITIAL APPROXIMATIONS

Exhibit 6.24. Roots Are: -2.33, .003 (2), i (2), 1.5i (2), -1.5i (2) 3i (3), -1-i (3)

```
POLYNOMIAL NUMBER 3
NUMBER OF INITIAL APPROXIMATIONS GIVEN. O
MAXIMUM NUMBER OF STERATIONS.
TEST FOR ZERO IN SUBROUTINE GCD. 0-100-02
TEST FOR CONVERGENCE.
                         0+100-09
TEST FOR ZERO IN SUBROUTINE QUAD. 0.100-19
RADIUS TO START SEARCH.
                         0.000 00
RADIUS TO END SEARCH.
                         0.000 00
THE DEGREE OF PIXI IS 8 THE COEFFICIENTS ARE
NO ROOTS OF MULTIPLICITY 1
THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF PIX) WHICH HAVE MULTIPLICITY 2
G(1 ) = -0.99999999999464980 00 + -0.1999999999964680 01 [
                                                                      INITIAL APPROXIMATION
ROOTS OF GIXE
```

REPEATED USE OF THE GREATEST COMMON DIVISOR AND MULLERS METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIALS

Exhibit 6.25.

0.4829629115656279D OD + 0.1294095284438187D OD I

ROOT( 1) = 0.9999999999464980 00 + 0.199999999946480 01 [

ROOTS OF PIXE

MULTIPLICITIES

INITIAL APPROXIMATION

IN THE ATTEMPT TO IMPROVE ACCURACY, ROUTE 21 = 0.20000000000025320 01 . 0.2000000000000016530 01 I

THE PRESENT APPROXIMATION IS 0.20019999742533280 01 + 0.20019999742524480 01 1

0.48296291156562790 00 + 0.12940952844381870 00 1

NOT ALL ROOTS OF THE ABOVE POLYNOMIAL, G. WERE FOUND

DID NOT CONVERGE AFTER 200 ITERATIONS

Exhibit 6.25. Roots Are: 2+2i (3), 1+2i (2), -1+.5i (3)

REPEATED USE OF THE GREATEST COMMON DIVISOR AND MULLERS METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIALS

```
NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR ZERO IN SUBROUTINE GCD. 0.100-09
TEST FOR ZERO IN SUBROUTINE QUAD. 0.100-19
TEST FOR ZERO IN SUBROUTINE QUAD. 0.000-00
RADIUS TO START SEARCH. 0.000 00
```

THE DEGREE OF PIX 15 12 THE COEFFICIENTS ARE

NO ROOTS OF MULTIPLICITY 1

NO ROOTS OF MULTIPLICITY .Z

Exhibit 6.26.

```
NO ROOTS OF MULTIPLICITY 3
NO ROOTS OF MULTIPLICITY 4
NO ROOTS OF MULTIPLICITY 5
THE FOLLOWING POLYNOMIAL. GIXI. CONTAINS ALL THE ROOTS OF PIX) WHICH HAVE MULTIPLICITY 6
INITIAL APPROXIMATION
ROOTS OF GIX)
RODT( 1) = 0.1000000000000033D 01 + 0.999999999999999707D 00 1
                                                                   0.48296291156562790 00 + 0.12940952844381870 00 I
SOLVED BY DIRECT METHOD
                                                     MULTIPLICITIES
                                                                               INITIAL APPROXIMATION
ADOTS OF PIX)
ROOT( 1) = 0.1000000000000000033D 01 + 0.999999999999707D 00 I ROOT( 2) = 0.10000000000000033D 01 + -0.999999999999707D 00 I
                                                                   0.4829629115656279D 00 + 0.1294095284438187D 00 I
                                                                   NO INITIAL APPROXIMATIONS
```

Exhibit 6.26. Roots Are: 1+i (6), 1-i (6)

## REFERENCES

- 1. S. D. Conte, <u>Elementary Numerical Analysis</u>, McGraw Hill, New York, 1965.
- 2. Peter Henrici, <u>Elements of Numerical Analysis</u>, John Wiley and Sons, Inc., New York, 1964.
- 3. Thomas R. McCalla, <u>Introduction to Numerical Methods and FORTRAN Programming</u>, John Wiley and Sons, Inc., New York, 1967.
- 4. David E. Muller, A method for solving algebraic equations using an automatic computer, Math. Tables and Aids to Comp., 10 (1956), 208-215.

#### APPENDIX A

## SPECIAL FEATURES OF NEWTON'S AND MULLER'S PROGRAMS

Several special features have been provided in each program as an aid to the user and to improve accuracy of the results. These are explained and illustrated below.\*

## 1. Generating Approximations

If the user does not have initial approximations available, sub-routine GENAPP can systematically generate, for an  $N^{th}$  degree polynomial, N initial approximations of increasing magnitude, beginning with the magnitude specified by XSTART. If XSTART is 0., XSTART is automatically initialized to 0.5 to avoid the approximation 0. + 0.i. The approximations are generated according to the formula:

$$X_{K} = (XSTART + 0.5K) (Cos \beta + i Sin \beta)$$

where

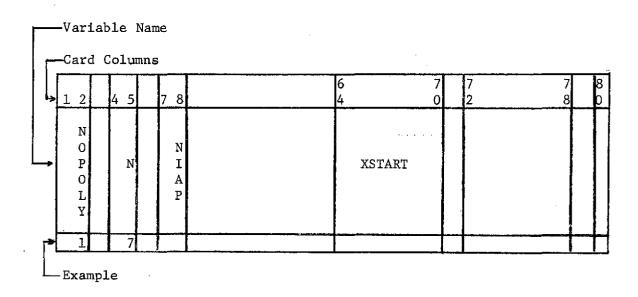
$$\beta = \frac{\Pi}{12} + K \frac{\Pi}{6}, K = 0,1,2,...$$

To accomplish this, the user defined the number of initial approximations to be read (NIAP) on the control card to be zero (0) or these

<sup>\*</sup>These illustrations are representative of Newton's method in double precision. The control cards for Muller's method are similarly prepared.

columns (7-8) may be left blank. If XSTART is left blank, it is interpreted as 0.

For example, a portion of a control card which generates initial approximations beginning at the origin for a seventh degree polynomial is shown in Example A.1.



Example A.1

The approximations are generated in a spiral configuration as illustrated in Figure A.1. Exhibit 6.1 is an example of output resulting from generated approximations.

Example A.2 shows a portion of a control card which generated initial approximations beginning at a magnitude of 25.0 for a sixth degree polynomial.

1 2	4 5	78	6 7 4 0	7 2	7 8	8 0
N O P O L Y	N	N I A P	XSTART			
1	6		2.5D+01			

Example A.2

Note that if the approximations are generated beginning at the origin, the order in which the roots are found will probably be of increasing magnitude. Roots obtained in this way are usually more accurate.

## 2. Altering Approximations

If an initial approximation,  $X_0$ , does not produce convergence to a zero within the maximum number of iterations, it is systematically altered a maximum of five times until convergence is possibly obtained according to the following formulas:

If the number of the alteration is odd: (j = 1,3)

$$X_{j+1} = |X_0|$$
 (Cos  $\beta + i$  Sin  $\beta$ ) where

$$\beta = \text{Tan}^{-1} \frac{\text{Im } X_0}{\text{Re } X_0} + K \frac{\Pi}{3}$$
;  $K = 1 \text{ if } j = 1, 2 \text{ if } j = 3.$ 

If the number of the alteration is even: (j = 0,2,4)

$$X_{j+1} = -X_{j}$$

Each altered approximation is then taken as a starting approximation. Each initial or altered approximation which does not produce convergence is printed as in Exhibit A.1. If none of the six starting approximations produce convergence, the next initial approximation is taken, and the process repeated. The six approximations are spaced 60 degrees apart on a circle of radius  $|X_0|$  centered at the origin as illustrated in Figure A.2.

# 3. Searching the Complex Plane

By use of initial approximations and the altering technique, any region of the complex plane in the form of an annulus centered at the origin can be searched for roots. This procedure can be accomplished in two ways.

The first way is more versatile but requires more effort on the part of the user. Specifically selected initial approximations can be used to define particular regions to be searched. For example, if the roots of a particular polynomial are known to have magnitudes between 20 and 40, an annulus of inner radius 20 and outer radius 40 could be searched by using the initial approximations  $20 \cdot + i$ ,  $23 \cdot + i$ ,  $26 \cdot + i$ ,  $29 \cdot + i$ ,  $32 \cdot + i$ ,  $35 \cdot + i$ ,  $38 \cdot + i$ ,  $40 \cdot + i$ .

By generating initial approximations internally, the program can search an annulus centered at the origin of inner radius XSTART and outer radius XEND. Values for XSTART and XEND are supplied on the control card by the user. Example A.3 shows a portion of a control card to search the above annulus of inner radius 20.0 and outer radius 40.0.

1 2	2	4 5	7 8	6 7 4 0	7 7 2 8	8 0
1 (	N D P D L	N	N A P	XSTART	XEND	
	1	7		2.0D+01	4.0D+01	

Example A.3

Note that since not less than N initial approximations can be generated at one time, the outer radius of the annulus actually searched may be greater than XEND but not greater than XEND  $\pm$  .5N.

Example A.4 shows a control card to search a circle of radius 15.

Ġ,

1 2	4 5	 7 8	6 4	7	7 7 2 8		8 0
N O P O L Y	N	N I A P	XSTART		XEND		
2	7				1.5D+01	<del>                                     </del>	H

Example A.4

Figure A.3 shows the distribution of initial and altered approximations for an annulus of width 2 and inner radius a.



## 4. Improving Zeros Found

After the zeros of a polynomial are found, they are printed under the heading "Before the Attempt to Improve Accuracy." They are then used as initial approximations with Newton's (Muller's) method applied each time to the full (undeflated) polynomial. In most cases, zeros that have lost accuracy due to roundoff error in the deflation process are improved. The improved zeros are then printed under the heading "After the Attempt to Improve Accuracy." Since each root is used as an approximation to the original (undeflated) polynomial, it is possible that the root may converge to an entirely different root. This is especially true where several zeros are close together. Therefore, the user should check both lists of zeros to determine whether or not this has occurred. See Exhibit 6.4.

# 5. Solving Quadratic Polynomial

After N-2 roots of an N<sup>th</sup> degree polynomial have been extracted, the remaining quadratic,  $ax^2 + bx + c$ , is solved using the quadratic formula

$$X = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

for the two remaining roots. These are indicated by the words "Solved By Direct Method" in the initial approximation column. If only a polynomial of degree 1 is to be solved, the solution is found directly as (X - C) = 0 implies X - C.

## 6. Missing Roots

If not all N roots of an N<sup>th</sup> degree polynomial are found, the coefficients of the remaining deflated polynomial are printed under the heading "Coefficients of Deflated Polynomial For Which No Zeros Were Found." The user may then work with this polynomial in an attempt to find the remaining roots. The coefficient of the highest degree term will be printed first (Exhibit A.2).

#### 7. Miscellaneous

By using various combinations of values for NIAP, XSTART, and XEND, the user has several options available as illustrated below.

Example A.5 shows the control card for a seventh degree polynomial. Three initial approximations are supplied by the user. At most three distinct roots will be found and the remaining deflated polynomial will be printed (Exhibit A.2).

1 -2	4 5	7 8	6 7 4 0	7 2	7 8	8 0
M C F C	N	N I A P	XSTART		XEND	
]	7	3				П

Example A.5

Note that if several roots are known to the user, they may be "divided out" of the original polynomial by using this procedure.

Example A.6 indicates that 2 initial approximations are supplied by the user to a  $7^{th}$  degree polynomial. After these approximations are used the circle of radius 15 will be searched for the remaining roots.

1	2	4 5	7 8	 6 7 4 0	7 7 2 8	8 0
	N O P O L Y	N	N I A P	XSTART	XEND	
	1	7	2		1.5D+01	

Example A.6

By defining XSTART between 0. and 15. an annulus instead of the circle will be searched (Exhibit A.3).

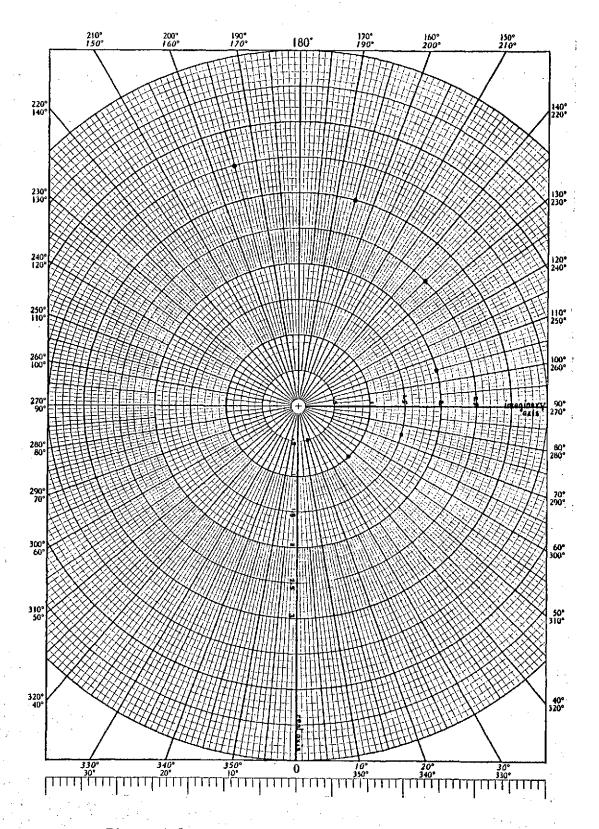


Figure A.1. Generating Initial Approximations

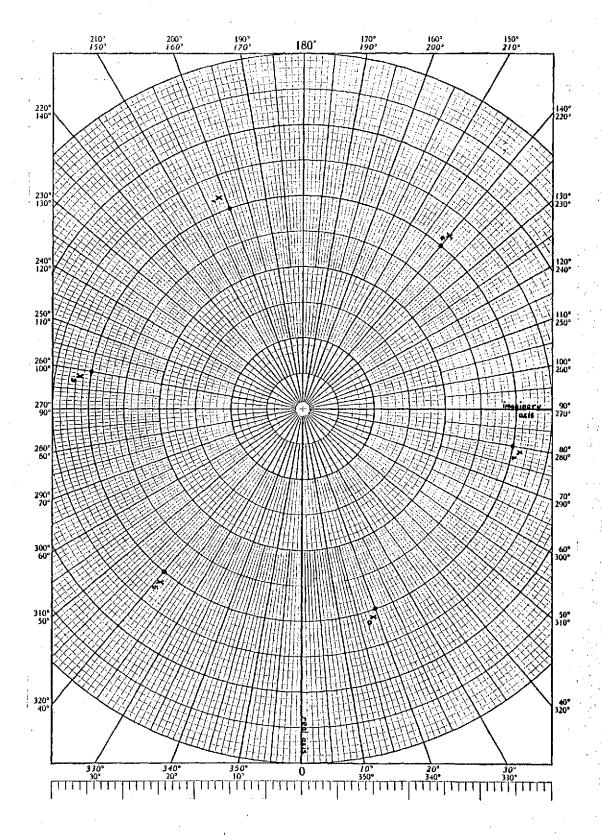


Figure A.2. Altering Approximations

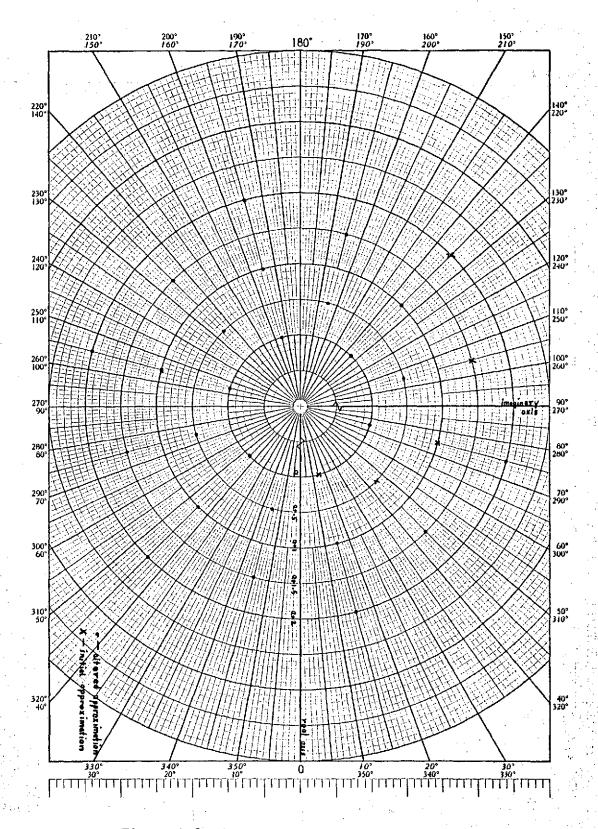


Figure A.3. Distribution of Approximations

# NEWTONS METHOD TO FIND ZERDS OF POLYNOMIALS POLYNOMIAL NUMBER 2 OF DEGREE 3

#### THE COEFFICIENTS OF PIXT ARE

NUMBER OF INITIAL APPROXIMATIONS	GIVEN. (
MAXIMUM NUMBER OF FTERATIONS.	
TEST FOR CONVERGENCE.	0.10D-0
TEST FOR MULTIPLICITIES.	0-100-0
RADIUS TO START SEARCH.	0.000 0
RADIUS TO END SEARCH.	D_00D D

#### NO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AFTER 3 ITERATIONS.

```
INITIAL APPROXIMATION
 0.48296291156562790 00 + 0.12940952844381870 00 1
-0.48296291156562790 00 + -0.12940952844381870 00 I
                                                                                          ALTERED APPROXIMATION
 0.1294094930884686D 00 + 0.4829629210390644D 00 1
                                                                                          ALTERED APPROXIMATION
-0.1294094930884666D 00 + -0.4829629210390644D 00 T -0.3535534294161402D 00 + 0.353553351770403DD 00 T
                                                                                          ALTERED APPROXIMATION
                                                                                          ALTERED APPROXIMATION
  0.35355342941614020 00 + -0.35355335177040300 00 1
                                                                                          ALTERED APPROXIMATION
 0.70710675530463460 00 + 0.70710680706845950 00 1
                                                                                          INITIAL APPROXIMATION
-0.70710675530463460 00 + -0.70710880706845950 00 I
-0.25881912758833590 00 + 0.96592580418437740 00 I
0.25881912759833590 00 + 0.96592580418437740 00 I
-0.96592580102499680 00 + 0.25881891546623570 00 I
                                                                                          ALTERED APPROXIMATION
                                                                                          ALTERED APPROXIMATION
                                                                                          ALTERED APPROXIMATION
                                                                                          ALTERED APPROXIMATION
 D.9659258610249968D DD + -0.2588189154662357D CO I
                                                                                          ALTERED APPROXIMATION
  0.38822847926540560 00 + 0.14488887631171930 01 E
                                                                                           INITIAL APPROXIMATION
0.38822847926540560 00 + -0.14488867631171930 01 E

-0.1060660288248421D 01 + -0.10606600553112090 01 I

0.1060660288248421D 01 + -0.10606600553112090 01 I

-0.1448886677856240D 01 + -0.3882287974635502D 0D I

0.1448888677856240D 01 + 0.3882287974635502D 00 I
                                                                                          ALTERED APPROXIMATION
                                                                                          ALTERED APPROXIMATION
                                                                                          ALTERED APPROXIMATION
                                                                                          ALTERED APPROXIMATION
```

#### COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS MERE-FOUND

Exhibit A 1

# NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS POLYNOMIAL NUMBER 1 OF DEGREE 7

EXPERIENTS OF DEFLATED POLYMONIAL FOR MMICH NO ZERDS MERE FOUND

D1 11 = 0.1000000000000000000 D1 + 0.0000000000000000 D0 I

D1 21 = -0.200000000000000 D1 + 0.600000000000000 D0 I

D1 31 = -0.200000000000000 D2 + -0.1899999999999 D2 I

D1 41 = 0.410000000000000 D2 + -0.2200000000000 D2 I

D1 51 = 0.23000000000000000 D2 + -0.4200000000000 D0 Z

```
THE COEFFICIENTS OF PIXE ARE
 P( 5) = 0.70000000000000010 02 + 0.72300000000000000 03 |
P( 6) = -0.1624000000000000 04 + -0.69600000000000010 03 |
 Pt 71 - C.19220000000000000 04 + -C.18320000000000000 04 1
 NUMBER OF INITIAL APPROXIMATIONS GIVEN. 3
MAXIMUM NUMBER OF ITERATIONS.
                                 200
                             0.100-09
TEST FOR CONVERGENCE.
TEST FOR MULTIPLICITIES.
                             0-100-01
0-000 00
RADIUS TO START SEARCH.
RADIUS TO END SEARCH.
                             0.000 00
BEFORE THE ATTEMPT TO IMPROVE ACCURACY. THE ZEROS OF PIXI ARE
 ROOTS OF PIKE
                                                       PULTIPLICITIES
                                                                                 INITIAL APPROXIMATION
 -0.35000000000000000 01 + -0.3500000000000000 01 E
 #3717 2) = 0.200000000000000000 01 + 0.2000000000000000 01 1 R007( 3) = -0.9999999999999900 01 I
                                                                   0.25000000000000000 01 + 0.250000000000000000 01 I
                                                                   -0.15000000000000000 01 + -0.4500000000000000 01 1
AFTER THE ATTEMPT TO IMPROVE ACCURACY. THE ZEROS OF PIXE ARE
                                                       MATIPLICITIES
 ROOTS OF PIXE
                                                                                 INITIAL APPROXIMATION
 #0016 II = -0.29999999999999 DI + -0.30000000000000 DI 1
                                                                   -0.35000000000000000 01 + -0.350000000000000000 01 1
 0-250000000000000000 01 + 0.250000000000000000 01 1
```

Exhibit A.2. Roots Are: -1 - 4i, -2 - 3i, -3 - 3i, -1 - i, 2 + 2i, 4 - i, 2 - i

# MENTONS METHOD TO FIND ZEROS OF POLYNOMIALS POLYNOMIAL NUMBER 1 OF DEGREE 7

#### THE COEFFICIENTS OF PIXI ARE

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 2

MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.100-01
TEST FOR MULTIPLICITIES. 0.100-01
RADIUS TO START SEARCH. 0.700 01
RADIUS TO END SEARCM. 0.150 02

BEFORE THE ATTEMPT TO IMPROVE ACCURACY. THE ZEROS OF PIX! ARE

# ROOTS OF P(x) RUSTIPLICITIES RUSTIAL APPROXIMATION ROOTS OF P(x) ROOTS OF P(x

AFTER THE ATTEMPT TO EMPROVE ACCUALCY, THE ZEROS OF P(X) ARE

```
ROOTS OF P(x)

ROOTS
```

Exhibit A.3. Roots Are: -1 - 4i, -2 - 3i, -3 - 3i, -1 - i, 2 + 2i, 4 - i, 2 - i

#### APPENDIX B

## NEWTON'S METHOD

## 1. Use of the Program

A double precision FORTRAN IV program using Newton's method is presented here. Flow charts for this program are given in Figure B.6 while Table B.VIII gives a FORTRAN IV listing of this program. Single precision variables are listed in some of the tables. The single precision variables are used in the flow charts and the corresponding double precision variables can be obtained from the appropriate tables.

The program is designed to solve polynomials of degree 25 or less. Both the coefficient of the highest degree term and the constant coefficient should be non-zero. In order to solve polynomials of degree N, where N > 25, certain array dimensions must be changed. These are listed in Table B.I for the main program and subprograms in double precision.

. . . . . . . . .

### TABLE B.I

# PROGRAM CHANGES FOR SOLVING POLYNOMIALS OF DEGREE GREATER THAN 25 BY NEWTON'S METHOD

### Double Precision

Main Program

RA(N+1), VA(N+1)
RB(N+1), VB(N+1)
RC(N+1), VC(N+1)
RD(N+1), VD(N+1)
RCOEF(N+1), VCOEF(N+1)
MULT(N)
RXZERO(N), VXZERO(N)
RX(N), VX(N)
RXINIT(N), VXINIT(N)

### Subroutine HORNER

RA(N+1), VA(N+1) RB(N+1), VB(N+1) RC(N+1), VC(N+1)

### Subroutine BETTER

RXZERO(N), VXZERO(N) RX(N), VX(N) RA(N+1), VA(N+1) RCOEF(N+1), VCOEF(N+1) RC(N+1), VC(N+1) RB(N+1), VB(N+1)

Subroutine GENAPP

APPR(N), APPI(N)

Subroutine QUAD

UA(N+1), VA(N+1) UROOT(N), VROOT(N) MULTI(N)

Table B.II lists the system functions used in the program of Newton's method. In the table "d" denotes a double precision variable name.

TABLE B.II

#### SYSTEM FUNCTIONS USED IN NEWTON'S METHOD

### Double Precision

DABS(d) - obtain absolute value:

DCOS(d) - obtain cosine of angle

DSIN(d) - obtain sine of angle

 $\mathtt{DATAN2}(\mathsf{d}_1,\mathsf{d}_2) \qquad \mathtt{-} \ \mathtt{arctangent} \ \mathsf{of} \ \mathsf{d}_1/\mathsf{d}_2$ 

DSQRT(d) - square root

### 2. Input Data for Newton's Method

The input data for Newton's method is grouped into polynomial data sets. Each polynomial data set consists of the data for one and only one polynomial. As many polynomials as the user desires may be solved by placing the polynomial data sets one behind the other. Each polynomial data set consists of three kinds of information placed in the following order:

- 1. Control information.
- 2. Coefficients of the polynomial.
- Initial approximations. These may be omitted as described in Appendix A, § 1.

An end card follows the entire collection of data sets. It indicates that there is no more data to follow and terminates execution of the program. This information is displayed in Figure B.1 and described below. For the double precision data, the D-type specification should

be used. All data should be right justified. The recommendations given in Table B.III are those found to give best results on the IBM 360/50 computer which has a 32 bit word.

### Control Information

The control card is the first card of the polynomial data set and contains the information given in Table B.III. See Figure B.2.

TABLE B.III

CONTROL DATA FOR NEWTON'S METHOD

Variable Name	Card Columns	Description
NOPOLY	c.c. 1-2	Number of the polynomial. Integer. Right justified.
N	c.c. 4-5	Degree of the polynomial. Integer. Right justified.
NIAP	c.c. 7-8	Number of initial approximations to be read. Integer. If no approximations are given, this should be left blank.
MAX	c.c. 19-21	Maximum number of iterations. Integer. Right justified. 200 is recommended.
EPSCNV	c.c. 30-35	Convergence requirement. Double precision. 1.D-10 is recommended.

TABLE B.III (Continued)

Variable Name	Card Columns	Description
EPSQ·	c.c. 37-42	Tolerance check for zero (0) in subroutine QUAD. Double precision. Right justify.
EPSMUL	c.c. 44-49	Multiplicity requirement. Double precision. Right justify. 1.D-02 is recommended.
XSTART	c.c. 64-70	Magnitude at which to begin generating initial approximations.  Double precision.  Right justify.  This is a special feature of the program and may be omitted.
XEND	c.c. 72-78	Magnitude at which to end the generating of initial approximations.  Double precision.  Right justify.  This is a special feature of the program and may be omitted.
KCHECK	c.c. 80	This should be left blank.

### Coefficients of the Polynomial

The coefficient cards follow the control card. For an N<sup>th</sup> degree polynomial, N+1 coefficients must be entered one per card. The coefficient of the highest degree term is entered first. For example, if the polynomial  $X^5 + 3X^4 + 2X + 5$  were to be solved, the order in which the coefficients would be entered is: 1, 3, 0, 0, 2, 5. Each

coefficient is entered, one per card, as described in Table B.IV and illustrated in Figure B.3.

TABLE B.IV

COEFFICIENT DATA FOR NEWTON'S METHOD

Variable Name	Card Columns	Description
RA (A in single precision)	c.c. 1-30	Real part of complex coefficient. Double precision. Right justify. If none, leave blank or enter 0.0D00.
VA (A in single precision)	c.c. 31-60	Imaginary part of complex coefficient. Double precision. Right justify. If none, leave blank or enter 0.0000.

### Initial Approximations

The initial approximation cards follow the set of coefficient cards.

The number of initial approximations read must be the number specified on the control card and are entered, one per card, as given in Table B.V and illustrated in Figure B.4.

TABLE B.V

INITIAL APPROXIMATION DATA FOR NEWTON'S METHOD

<u>Variable Name</u>	Card Columns	Description		
RXZERO (XZERO in single precision)	c.c. 1-30	Real part of complex number. Double precision. Right justify. If none, leave blank or enter 0.0D00.		
VXZERO (XZERO in single precision)	c.c. 31-60	Imaginary part of complex number. Double precision. Right justify. If none, leave blank or enter 0.0000.		

### End Card

The end card is the last card of the input data to the program. It indicates that there is no more data to be read. When this card is read, program execution is terminated. This card is described in Table B.VI and illustrated in Figure B.5.

TABLE B.VI

DATA TO END EXECUTION OF NEWTON'S METHOD

Variable Name	Card Columns	Description
KCHECK	c.c. 80	Must contain the number 1. Integer.

### 3. Variables Used in Newton's Method

The definitions of the major variables used in Newton's method are given in Table B.VII. The symbols used to indicate type are:

R - real variable

I - integer variable

C - complex variable

D - double precision

L - logical variable

A - alphanumeric variable

When two variables are listed, the one on the left is the real part of the corresponding single precision complex variable; the one on the right is the imaginary part. The symbols used to indicate disposition are:

E - entered

R - returned

ECR - entered, changed, and returned

C - variable in common

### 4. Description of Program Output

The output from Newton's method programs consist of the following information.

The number and degree of the polynomial are printed in the heading (Exhibit 6.1).

The coefficients are printed under the heading "THE COEFFICIENTS OF P(X) ARE." The coefficient of the highest degree term is listed first (Exhibit 6.1).

As an aid to ensure the control information is correct, the number of initial approximations given, maximum number of iterations, test for convergence, test for multiplicaties, radius to start search, and radius to end search are printed as read from the control card (Exhibit 6.1).

The zeros found before and after the attempt to improve accuracy are printed. See Appendix A, § 4 for further explanation (Exhibit 6.1).

If not all zeros of the polynomial are found, the coefficients of the remaining unsolved polynomial will be printed, with coefficient of highest degree term first, under the heading "COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND." See Appendix A, § 6. This is illustrated in Exhibit A.2.

The multiplicity of each zero is given under the title "MULTIPLI-CITIES" (Exhibit 6.1).

The initial approximation producing convergence to a root is printed to the right of the corresponding root and headed by "INITIAL APPROXIMATION." The initial approximations may be those supplied by the user, or generated by the program, or a combination of both (Exhibit A.3). See Appendix A, § 1 and § 2 for discussion of approximations. The message "SOLVED BY DIRECT METHOD" indicates that the corresponding root or roots was obtained by Subroutine QUAD. See Appendix A, § 5.

If an approximation does not produce convergence within the maximum number of iterations, it is printed under the heading "NO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AFTER XXX ITERATIONS." XXX is replaced by the maximum number of iterations. The type of the approximation, that is, initial approximation or altered approximations is given (Exhibit A.1). See Appendix A, § 1 and § 2 for discussion of approximations.

### 5. Informative and Error Messages

The output may contain informative or error messages. These are intended as an aid to the user and are described as follows:

"IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(X) = YYY DID NOT CONVERGE
THE PRESENT APPROXIMATION AFTER ZZZ ITERATIONS IS PRINTED BELOW." X is
the number of the zero, YYY is the value of the zero before the attempt
to improve accuracy, ZZZ is the maximum number of iterations. This
message indicates that a zero found before attempting to improve
accuracy did not converge sufficiently when being used as an initial
approximation on the full (undeflated) polynomial. The current approximation is printed in the list of improved zeros. In many cases, this
failure to converge is a result of an ill-conditioned polynomial and
this current approximation of the root may be better than its approximation before the attempt to improve accuracy. In most cases, the
polynomial from which this root was first extracted had fewer multiple
roots, due to deflations, than the original polynomial.

"THE VALUE OF THE DERIVATIVE AT XO = XXX IS ZERO."

This message is printed as a result of the value of the derivative of the original polynomial at an approximation, XXX, being zero (0). It occurred in the attempt to improve the accuracy of a zero. The previous message is then printed.

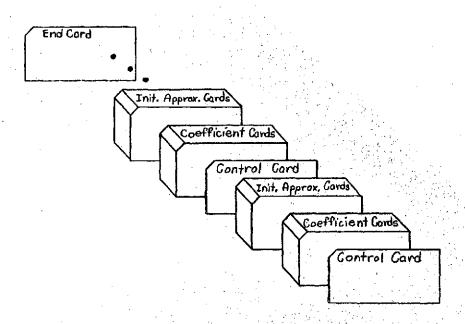


Figure B.1. Sequence of Input Data for Newton's Method

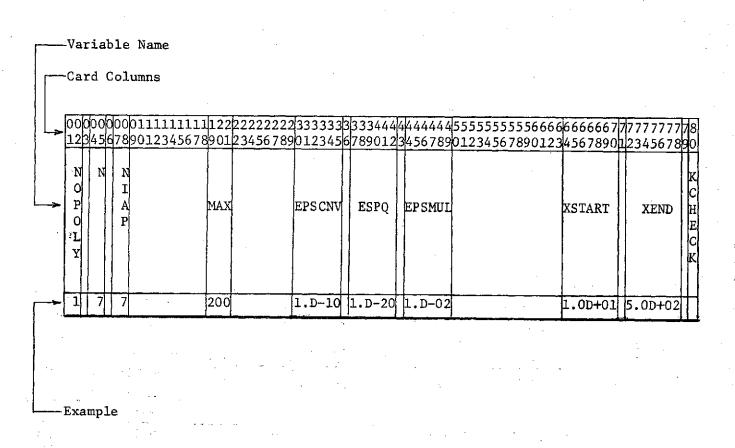


Figure B.2. Control Card for Newton's Method

0000000011111111112222222223	333333333444444444455555555555	6666666677777777778
123456789012345678901234567890	<u> 123456789012345678901234567890</u>	12345678901234567890
A (RA)	A (VA)	
()	ZZ (VZ)	
0.621735D+01	-0.132714D-02	
		l

Figure B.3. Coefficient Card for Newton's Method

00000000011111111112222222223333 12345678901234567890123456789012	3333333444444444455555555556 8456789012345678901234567890	6666666667777777777 12345678901234567890
XZERO (RXZERO)	XZERO (VXZERO)	
0.15D+01	-0.25D-00	

Figure B.4.Initial Approximation Card for Newton's Method

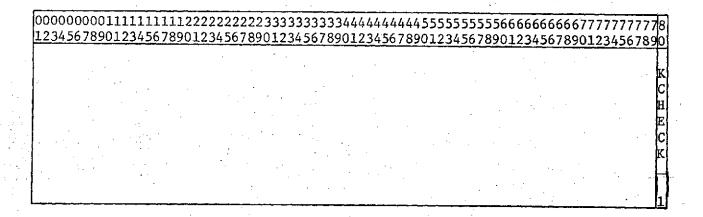


Figure B.5. End Card for Newton's Method

TABLE B. VII

VARIABLES USED IN NEWTON'S METHOD

Single Precision Variable Type	Double Precision Variable Type		Description
		Main	Program
NOPOLY I N I NIAP I MAX I EPSCNV R EPSMUL R EPSQ R XSTART R XEND R KCHECK I	NOPOLY I N I NIAP I MAX I EPSCNV D EPSMUL D EPSQ D XSTART D XEND D KCHECK I		Number of the polynomial Degree of the polynomial Number of initial approximations to be read Maximum number of iterations to be performed Tolerance check for convergence Tolerance check for multiplicities Tolerance check for zero in subroutine QUAD Magnitude from which to begin the search for zeros Magnitude to end the search for zeros Program Control. When KCHECK = 1, program will
NA I A C	NA I RA,VA D		terminate execution.  Number of coefficients or original polynomial  Array containing the coefficients of original polynomial P(X)
NDEF I L I ITER I NROOT I	NDEF L ITER INROOT I		Degree of current deflated polynomial Counter for number of initial approximations used Counter for number of iterations Counter for number of roots found (counting multiplicities)
IALTER	IALTER		Counter for number of alterations of each initial approximation
ITIME I K I ND I	ITIME I K I ND I		Program control Counter for number of distinct roots found Program control & number of coefficient of deflated polynomial for which no zeros were found

TABLE B. VII (Continued)

•					
Single Pre	cision	Double Precisi	ion :	Disposition	
Variable	Type			of Argument	Description
				<u> </u>	Description
XO	С	RXO, VXO	D .	•	Current approximation (Xn) to root
COEF	С	RCOEF, VCOEF	<b>D</b> ·		Working array containing coefficients of current
1		n			deflated polynomial
DPX	Ç		D		Derivative of P(X) at some value X
PX	C	,	D		Value of P(X) at some point X
XZERO	C	· -	D		Array containing the initial approximations
		VXZERO			,
XNEW	C	RXNEW, VXNEW	D		New approximation $(X_{n+1})$ obtained from old approximation
					(X <sub>n</sub> ) by Newton's Algorithm
KANS	Ţ	KANS	I		KANS = 1 implies convergence, KANS = 0 implies no
		·		•	convergence
MULT	Ι	MULT	I		Array containing the number of multiplicities of each
•					root
X	. C	RX,VX	-	· · ·	Array containing the zeros of P(X)
XINIT	С	RXINIT, D	)		Array containing the initial or altered approximations
NITTA	-	VXINIT	_		which produced convergence to each root
NUM B	I		Z ·		Number of coefficients of current deflated polynomial
D	C	RB, VB	)		Array containing the coefficients of newly deflated
IROOT	I	TDOOM		•	polynomial
TROOT	T.	IROOT			Number of distinct roots found by Newton's method, i.e.
D .	C	מזו מת		•	not solved for directly by subroutine QUAD
ע .	C	RD, VD D	, .		Array containing the coefficients of deflated polynomial
101	I	I01 I	. :		for which no zeros were found
102	Ī	101 I			Unit number of input device
C C	C				Unit number of output device
U		RC,VC D	1		Array containing sequence of values leading to the
EPSCHK	R	EPSCHK D	÷		derivative
THE O'CLES	K	EPSCHK D	1		Current tolerance for checking convergence or
			» ·		multiplicity

TABLE B. VII (Continued)

Single Pa Variable	recision Type	Double Preci Variable	ision Type	Disposition of Argument	Description
				Subr	outine HORNER
A	С	RA, VA	<b>D</b>	E	Array of coefficients of polynomial
В	С	RB, VB	D	R ·	Array of coefficients of deflated polynomial
NDEF	. I	NDEF	I	E	Degree of polynomial
NUM	I	NUM	Ī		Number of coefficients of polynomial
X0	Ç	RXO,VXO	D.	E	Point $(X_n)$ at which to evaluate the polynomial and its
		·	_		derivative. Also current approximation $(X_{n+1})$ used
•				•	to deflate the polynomial
PX	C	RPX, VPX	D	R	Value of polynomial at X <sub>n</sub>
DPX	С	RDPX, VDPX	. d	R	Value of the derivative of polynomial at $X_n$
C	С	RC, VC	D	R	Array of containing sequence of values leading to the
		,		•	derivative
•		•			
				Subrou	tine NEWTON
PX	C	RPX, VPX	D	E	Value of polynomial at X <sub>D</sub>
DPX	C	RDPX, VDPX	D	E	Derivative of polynomial at Xn
X0	C	RXO,VXO	מ	E	Current approximation (Xn) to root
XNEW	С	RXNEW, VXNEW	D	R	New approximation (X ,) to root
•			<i>D</i>		n+1, to 100t
			i	Subro	utine CHECK
EPSLON	R	EPS	D	C.	Tolomonos for accompany
PX	ĉ	RPX, VPX	D	E	Tolerance for convergence or multiplicity check
DPX	Č	RDPX,VDPX	D	Ē	Value of $P(X)$ at $X_n$
XO	Č	RXO,VXO	D	E	Derivative of P(X) at X <sub>n</sub>
102	ĭ	102	I	C	Current approximations (X <sub>n+1</sub> ) to root
KANS	ī	KANS	I	R	Unit number of output device
	-	**************************************	<b>-1-</b>	A	KANS = 1 implies convergence, KANS = 0 implies no convergence

TABLE B. VII (Continued)

Single Pre Variable	cision Type	Double Precisio Variable Typ	-	Description
			Subro	outine BETTER
102	I	102 I	C	Unit number of output device
XZERO	С	RXZERO, D VXZERO	E	Array of approximations
X	С	RX,VX D	ECR	Array of roots
A	С	RA, VA D	E	Coefficients of original (undeflated) polynomial, P(X)
COEF	С	RCOEF, VCOEF D		Working array for coefficients of polynomial
NA	I	NA I	E	Number of coefficients of original polynomial
XO	С	RXO,VXO D		Current approximation (Xn) to root
DPX	С	RDPX, VDPX D		Derivative of $P(X)$ at $X_n$
PX	С	RPX, VPX D	•	Value of P(X) at X <sub>n</sub>
KANS	I	KANS I	·.	KANS = 1 implies convergence; KANS = 0 implies no convergence
ITER	I	ITER		Counter for number of iterations
XNEW	C .	RXNEW, VXNEW D		New approximation (Xn+1) to root
NN	I	NN		Degree of polynomial
<b>.</b> C	С	RC,VC D	E	Array containing the sequence of values leading to the derivative
K	I	K	E	Number of distinct roots of P(X) found
N	I	N	E	Degree of polynomial P(X)
В	С	RB,VB D	E	Array of coefficients of deflated polynomial
MAX	I .	MAX	C	Maximum number of iterations permitted
EPSCHK	R	EPS D	¨ C	Tolerance for checking convergence
			Subro	outine GENAPP
APP	С	APPR, APPI D	R	Array containing initial approximations
NAPP	Ĭ	NAPP I	E	Number of initial approximations to be generated
·	_		<del>-</del>	

TABLE B. VII (Continued)

Single Pre	cision	Double Prec	ision	Disposition	
Variable	Type	Variable	Type	of Argument	Description
XSTART	R	XSTART	, D	ECR	Magnitude at which to begin generating approximations;
BETA	R	BETA	D	•	also magnitude of the approximation being generated Argument of the complex approximation being generated
U.	R	APPR(I)	D	*	Real part of complex approximation being generated
V	R	APPI(I)	D	,	Imaginary part. of complex approximation
				Subro	utine ALTER
XOLD	C	XOLDR, XOLDI	D	ECR	Old approximation to be altered to new approximation
NALTER	I	NALTER	ī	ECR	Number of alterations performed on an initial approximation
ITIME	I	ITIME	I	E	Program control
MAX	I	MAX	I	С	Maximum number of iterations permitted
Y	R	XOLDI	Ď		Imaginary part of original initial approximation (unaltered)
X	R	XOLDR	Ď		
R	R	R	D		Real part of original unaltered initial approximation
BETA	R	BETA	D	•	Magnitude of original unaltered initial approximation Argument of new approximation
XOLDR	R	XOLDR	D		Real part of new approximation
XOLDI	R	XOLDI	D		Transparts next of any approximation
102	I	102	T I	С	Imaginary part of new approximation Unit number of output device
	-				
				Subro	outine QUAD
A	С	UA,VA	D	E	Coefficients of polynomial to be solved
NA	I	NA	I	E	Degree of polynomial
ROOT	С	UROOT, VROOT	D	ECR	Array of roots of P(X) (original polynomial)
NROOT	I	NROOT	I	ECR	Number of distinct roots of P(X) (the original
				,	polynomial)

TABLE B. VII (Continued)

Single Prec Variable	ision Type	Double Precis: Variable T		sposition Argument	Description
MULTI EPST DISC	I R C	MULTI EPST UDISC,VDISC	I D D	ECR E	Array containing multiplicaties of each root Tolerance check for the number zero Value of the discriminate (b2 - 4ac) of Quadratic
				Subrou	tine COMSQT
		UX,VX UY,VY	D D	E R	Complex number for which the square root is desired Square root of the complex number

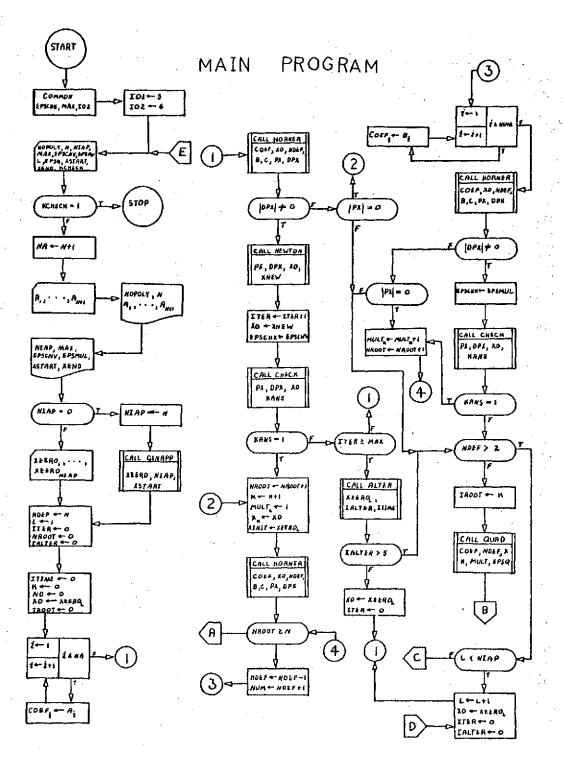


Figure B. 6. Flow Charts for Newton's Method

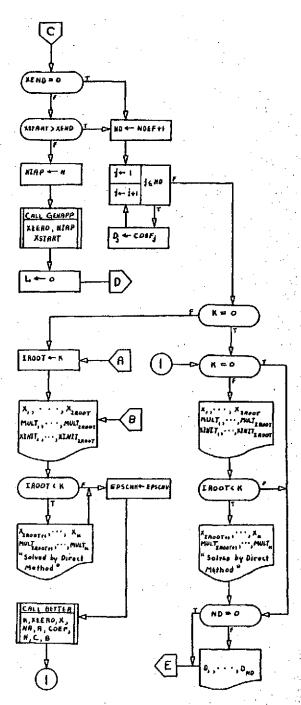


Figure B.6. (Continued)

BETTER

CHECK

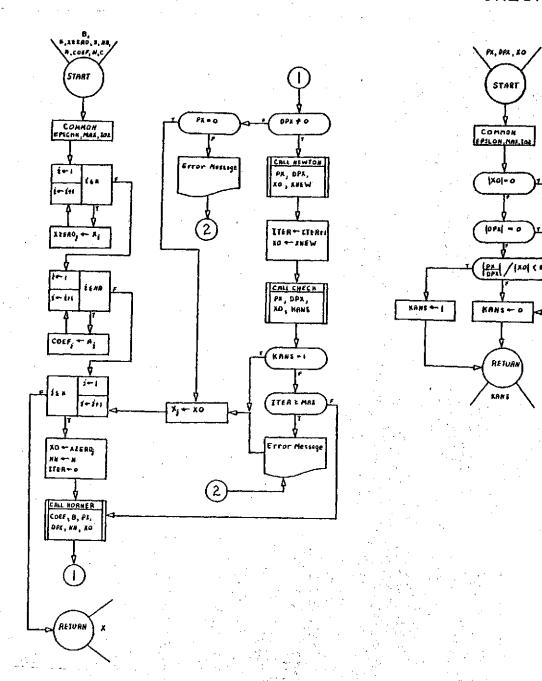
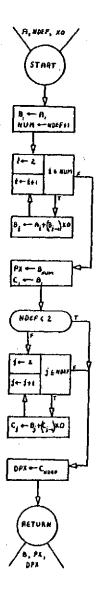


Figure B.6. (Continued)

## HORNER

## NEWTON



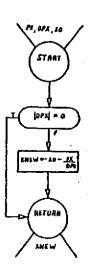
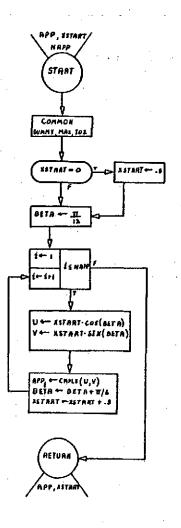


Figure B.6. (Continued)

### GENAPP

### ALTER



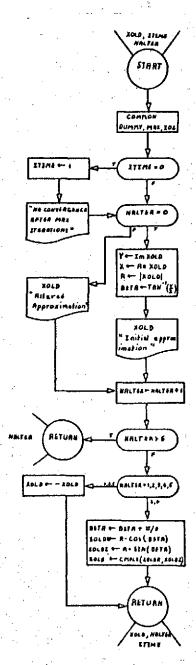
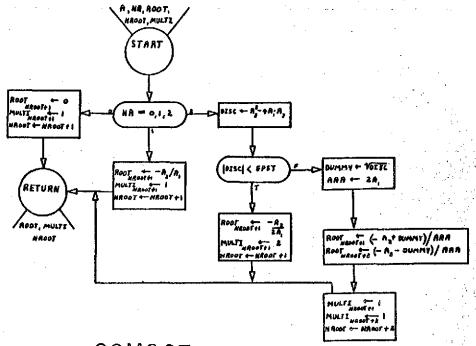


Figure B.6. (Continued)

### QUAD



### COMSQT

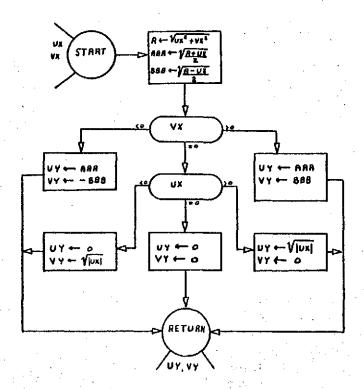


Figure B.6. (Continued)

#### TABLE B. VIII

#### PROGRAM FOR NEWTON'S METHOD

```
C
                                  20000
                                             * DOUBLE PRECISION PROGRAM FOR NEWTON'S METHOD
                                             * NEHTONS METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A
                                             * POLYNOMIAL OF MAXIMUM DEGREE 25 BY COMPUTING A SEQUENCE OF APPROX-

* IMATIONS CONVERGING TO A ZERO OF THE POLYNOMIAL USING THE ITERATION
                                             * FORMULA
                                                                                      x(N+1) = x(N) - P(x(N)) / P(x(N)).
                                  CCC
                                                   *************************
                                                  DOUBLE PRECISION RA, VA, RXZERO, VXZERO, RB, VB, RCOEF, VCOEF, RX, VX, RXINI
0001
                                                LT, VXINIT, RC, YC, RO, VD, RPX, RDPX, VDPX, RXNEW, VXNEW, RXO, VXO, EPSCHK, 2EPSCHY, EPSG, EPSMUL, XSTART, XEND, ABPX, ABDPX
DIMENSION RA(26), VA(26), RB(26), VB(26), RC(26), VC(26), RD(26), RD(26)
0002
                                                2XINIT(25), VXINIT(25)
                                                  COMMON EPSCHK. MAX. 102
0003
                                                                                                                                                                                                                                                      112
0004
                                                   101=5
                                                                                                                                                                                                                                                      116
0005
                                                   102=6
                                             1 READ(ID1,1000) NOPOLY, N, NIAP, MAX, EPSCNV, EPSQ, EPSMUL, XSTART, XEND, KC
0006
                                                THECK
                                                   IF(KCHECK.EQ.1) STOP
0007
                                                                                                                                                                                                                                                      130
8000
                                                  NA=N+1
0009
                                                   READ([D1,1010) [RA([],VA([),[=1,NA]
                                                                                                                                                                                                                                                      132
                                                  WRITE(102,1030) NOPOLY,N
WRITE(102,1040) (I,RA(I),VA(I),I=1,NA)
WRITE(102,2060)
 0010
0011
0012
                                                   WRITE(102,2000) NIAP
0013
                                                  WRITE(102,2010) MAX
WRITE(102,2010) MAX
WRITE(102,2020) EPSCNV
WRITE(102,2030) EPSMUL
WRITE(102,2040) XSTART
WRITE(102,2050) XEND
0014
 0015
 0016
 0017
 0018
 0019
                                                   IF(NIAP.NE.O) GO TO 3
 0020
                                                   NIAP=N
                                                   CALL GENAPP(RXZERO, VXZERO, NIAP, XSTART) .
 0021
                                                   GO TO 4
 0022
                                             3 READ(101,1020) (RXZERO(1), VXZERO(1), 1=1, NIAP)
 0023
 0024
                                                  NOEF=N
                                                                                                                                                                                                                                                      152
 0025
                                                   L=1
 0026
                                                   ITER=0
                                                                                                                                                                                                                                                      154
 0027
                                                   NROOT=0
                                                                                                                                                                                                                                                      160
                                                   IRDOT=0
 0028
                                                   ITIME=0
 0029
                                                   ND=0
 0030
 0031
                                                   IALTER=0
                                                                                                                                                                                                                                                      164
 0032
                                                   K≖O
                                                                                                                                                                                                                                                      168
                                                   RXD=RXZEROIL)
                                                                                                                                                                                                                                                       180
 0033
                                                   VXO=VXZERO(L)
                                                                                                                                                                                                                                                       161
 0034
                                                   DO 5 [=1,NA
RCOEF(1)=RA(1)
 0035
                                                                                                                                                                                                                                                       188
 0036
                                                                                                                                                                                                                                                      190
 0037
                                              5 VCDEF(I)=VA(I)
                                                                                                                                                                                                                                                       191
                                           10 CALL HORNERIRCOEF, VCOEF, RXO, VXO, NDEF, RB, VB, RC, VC, RPX, VPX, ROPX, VDPX
 0038
                                                   ABPX=DSQRT(RPX+RPX+VPX+VPX)
 0039
                                                   ABDPX=DSQRTIRDPX*RDPX+VDPX*VDPX1
 0040
```

0041		IF(ABDPX.NE.0.0) GO TO 20	
0042		IF(ABPX.EQ.0.0) GO TO TO	
0043		GO TO 110	
0044	20	CALL NEWTON (RPX.VPX.ROPX.VDPX.RXO.VXD.RXNEW.VXNEW)	
0045	20	ITER=ITER+1 200	•
0046		RXO=RXNEW 204	
0047		VXO=VXNEW 205	
0048		EPSCHK=EPSGNY	
0049	1	CALL CHECK(RPX, VPX, RDPX, VDPX, RXO, VXO, KANS)	
0050		IFIKANS.EQ.11 GD 70 70	,
0051		IF(ITER.GE.MAX) GO TO 40	
0052		GO TO 10 208	
0053	40	CALL ALTER(RXZERO(L), VXZERO(L), [ALTER, ITIME)	
0054		[F(IALTER.GT.5) GO TO 110	
0055		RXO=RX ZERO(L)	
0056		VXO=VXZERO(L)	
0057		1TER=0 244	
0058		60 TO 10 248	
0059	60	ND=NDEF+1	
0060		DO 65 J=1,ND	
0061		RD[J] #RCOEF(J)	
0062	65	VD(J)=VCOEF(J)	
0063		GO TO 140	
0064	70	NROOT=NROOT+1 268	
0065		K=K+1 : 272	
0066		MULT(K)=1 276	
0067		RX(K) = RXO 280	
0068		VX(K)=VXO 281	
0069		RXINIT(K)=RXZERO(L) 288	
0070		VXINIT(K)=VXZERO(L) 289	
0071		CALL HORNERIRCOEF, VCOEF, RXG, VXG, NDEF, R8, VB, RC, VC, RPX, VPX, RDPX, VDPX	
		1)	
0072	80	IFINROOT.GE.N) GO TO 147	
0073		NDEF=NDEF-1	
0074		NUM=NDEF+1	
0075		DO 105 [=1,NUM 294	
0076	100	RCOEF(I)=RB(I) 296 VCOEF(II=VB(II 297	
0077	. 102	CALL HORNER(RCDEF, VCOEF, RXO, VXO, NDEF, RB, VB, RC, VC, RPX, VPX, RDPX, VDPX	
0078		1)	
0079		ABPX=DSQRT(RPX*RPX+VPX*VPX)	٠,
0080	•	ABDPX=OSQRT(ROPX+RDPX+VDPX+VDPX)	
DOB1		IFIABDPX.NE.O.D) GO TO 107	
0082		[#(ABPX.EQ.0.0) GO TO 130	
0083		GO TO 110	
0084	107	CONTINUE	
0085	•	EPSCHK=EPSMUL	
0086	•	CALL CHECK(RPX, VPX, ROPX, VDPX, RXO, VXO, KANS)	
0087		1F(KANS.EQ.11 GO TO 130 300	
0088	110	IF(NDEF-GT-2) GO TO 113	
0089		IROOT=K	
0090		CALL QUAD(RCDEF, VCOEF, NDEF, RX, VX, K, NULT, EPSQ)	
0091		GO TO 150	
0092	113	IF(L.LT.NIAP) GO TO 115	
0093		IF(XEND.EQ.0.0) GO TO 60	
0094		IF(XSTART.GT.XEND) GO TO 60	
0095		NI AP=N	
0096		CALL GENAPP(RXZERO, VXZERO, NIAP, XSTART)	

```
0097
                     L=0
                115
009A
                     1 =1 +1
0099
                     RXD=RXZERO(L)
                                                                                                        312
0100
                     VXO=VXZERO(L)
                                                                                                        313
0101
                     1TER≈0
                                                                                                        316
0102
                     IALTER=0
                                                                                                        320
                GD TO 10
130 MULT(K)=MULT(K)+1
0103
                                                                                                        324
0104
                                                                                                        328
                     NROOT=NROOT+1
                                                                                                        332
0105
0106
                     GO TO 80
                                                                                                        336
0107
                140 IF(K.EQ.D) GO TO 160
0108
                 147 IRBOT=K
                150 WRITE(102,1025)
WRITE(102,1050)
0109
0110
                     WRITE(102,1060) (1,RX(1),VX(1),MULT(1),RXINIT(1),VXINIT(1), [=1, IRQ
0111
0112
0113
                     IF(IROOT.LT.K) WRITE(IO2,1062) (I,RX(I),VX(I),MULT(I),I=KKK,K)
0114
                     FPSCHK#FPSCNV
                     CALL BETTER(K, RXZERO, VXZERO, RX, VX, NA, RA, VA, RCOEF, VCOEF, N, RC, VC, RB,
0115
                    1 V8 }
0116
                160 IF(K.EQ.O) GO TO 170
0117
                     WRITE(102,1065)
0118
                     WRITE(102,1050)
                     WRITE(102,1060) (1,RX(1),VX(1),MULT(1),RXINIT(1),VX[NIT(1),I=1,IRQ
0119
                    1011
0120
                     KKK=IRGOT+1
                [F(1R007.17.K) WRITE(102,1062) (1,RX(1),VX(1),MULT(1),1=KKK,K) 170 [FIND.Eq.0] GO TO 1
0121
0122
0123
                     WRITE(102,1070)
                                                                                                        396
                     WRITE(102,1075) (J,RD(J),YD(J),J=1,ND1
0124
0125
                     GO TO 1
                                                                                                        444
               1000 FORMAT(3(12,1X),9X,13,8X,3(D6.0,1X),13X,2(D7.0,1X),[1)
0126
0127
               1010 FORMAT(2D30.0)
               1030 FORMAT(1H1.8x.43HNEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS/9x,18
0128
                    1HPOLYNOMIAL NUMBER .12.11H OF DEGREE .12.///1X.28HTHE COEFFICIENT
               25 DF P(X) ARE/)
1040 FORMAT(3X,2HP(,I2,4H) = ,D23.16,3H + ,D23.16,2H [)
0129
               1020 FORMAT(2D30.01
0130
               1025 FORMATI///1X,61HBEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS
0131
               10F P(X) ARE)
1050 FORMAT(///2X,13HROOTS OF P(X),52X,14HMULTIPLICITIES,17X,21HINITIAL
0132
                   1 APPROXIMATION//)
               1060 FORMAT (3x,5HROOT (,12,4H) = ,D23.16,3H + ,D23.16,2H I,7x,12,7x,D23.
0133
                    116,3H + ,D23,16,2H I)
0134
               1062 FORMAT(3X,5HRDOT(,12,4H) = ,D23,16,3H + ,D23,16,2H 1,7X,12,8X,23HS
               ICLVED BY DIRECT METHOD;
1065 FORMAT(///LX,61HAFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS
0135
               10F P(X) ARE)
10F P(X) ARE)
10F P(X) ARE)
0136
                    1EROS WERE FOUND/)
               1075 FORMAT(3X,2HD(,12,4H) = ,023.16,3H + ,D23.16,2H 1)
2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN.
2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,13)
0138
0139
               2020 FORMAT(1x,21HTEST FOR CONVERGENCE..13x,09.2)
2030 FORMAT(1x,24HTEST FOR MULTIPLICITIES.,10x,09.2)
0140
0141
0142
               2040 FORMAT(1x,23HRADIUS TO START SEARCH.,11x,D9.2)
0143
               2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,09.2)
0144
               2060 FORMAT (//1X)
0145
                     END
                                                                                                        450
```

```
0001
                        SUBROUTINE GENAPPIAPPIAPPINAPPIXSTARTI
                00000
                     * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE * DEGREE OF THE ORIGINAL POLYNOMIAL.
0002
                        DOUBLE PRECISION APPR, APPI, XSTART, DUMMY, BETA
                        DIMENSION APPRIZ51, APPIZ251
COMMON DUMMY, MAX, 102
0003
0004
                        IF(XSTART.EQ.J.0) XSTART=0.5
BETA=0.2617994
0005
0006
                        DO 10 [=1.NAPP
APPR(LI=XSTART*DCOS(BETA)
0007
8000
                        APPI(I)=XSTART*DSIN(BETA)
BETA=BETA+0.5235988
XSTART=XSTART+0.5
0009
0010
0011
0012
                        RETURN
0013
                        END
0001
                        SUBROUTINE ALTER (XOLDR, XOLDI, NALTER, ITIME)
                000000
                      * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT.
0002
                        DOUBLE PRECISION XOLDR, XOLDI, DUMMY, ABXOLD, BETA
0003
                        COMMON DUMMY, MAX, 102
                        IF(ITIME.NE.O) GO TO 5
0004
                     ITIME =1
WRITE(102,1010) MAX
5 IF(NALTER.EQ.O) GO TO 10
WRITE(102,1000) XOLOR,XOLOI
0005
0006
0007
8000
0009
                        GO TO 20
                    10 ABXOLD=DSQRT(XOLDR*XOLDR*XOLDI)
0010
                        BETA=DATAN2(XOLDI,XOLDR)
0011
0012
                        WRITE(102,1020) XOLOR, XOLDI
0013
                    20 NALTER=NALTER+L
0014
                        IFINALTER.GT.51 RETURN
                    GO TO (30,40,30,40,30),NALTER
30 XOLDR=-XOLDR
0015
0016
0.017
                        XOLDI =- XOLDI
0018
                        GO TO 50
0019
                        BETA=BETA+1.0471976
                        XOLDR=ABXOLD*DCOS(BETA)
0020
0021
                        XOLDI=ABXOLD*DSIN(BETA)
                    50 RETURN
0022
                 1000 FORMAT(1X,D23.16,3H + ,D23.16,2H 1,10X,21HALTERED APPROX(MATION)
1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
0023
0024
                       ITER ,13.12H ITERATIONS.//1
                 1020 FORMAT(/1X,023.16,3H + ,D23.16,2H 1,10x,21HINITIAL APPROXIMATION)
0025
                        END
0026
```

```
0001
                         -SUBROUTINE QUADILLA.VA.NA.URGOT.VROOT.NROOT.HULTI.EPST1
                 0000
                          SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR. SOLUTION OF THE QUADRATIC IS DONE USING THE QUADRATIC FORMULA.
                       *
                              ********************************
                         DOUBLE PRECISION UA, VA, UROOT, VROOT, BBB, UAAA, VAAA, UDISC, VDISC, UDUMM
LY, VDUMMY, ROUMMY, SDUMMY, EPST, UBBB, VBBB
DIMENSION UA(26), VA(26), UROOT(25), VROOT(25), HULTI(25)
0002
0003
                          IF(NA.EQ.2) GO TO 7
IF(NA.EQ.1) GO TO 5
URDOT(NROOT+1)=0.0
0004
0005
0006
                          VROOT (NROOT+1)=0.0
MULTI(NROOT+1)=1
0007
0008
                          NROOT=NROOT+1
0009
0010
                          GO TO 50
0011
                          888=UA(1)*UA(1)+VA(1)*VA(1)
                          UROOT(\ROOT+1)=(-UA(2)*UA(1)-VA(2)*VA(1))/BBB
VROOT(\ROOT+1)=(-VA(2)*UA(1)+UA(2)*VA(1))/BBB
0012
0013
                          MULTI(NROOT+1)=1
0014
                          NROOT=NROOT+1
0015
                          GG TO 50
UDISC={UA(2)*UA(2)-VA(2)*VA(2)}-(4.0*(UA(1)*UA(3)-VA(1)*VA(3))}
0016
                          VDISC=(VA(2)*VA(2)+VA(2)+VA(2))-(4.0*(VA(1)*VA(3)+VA(1)*VA(3)))
BBB=DSQRT(UDISC*VDISC*VDISC*VDISC)
IF(BBB.LI.EPST) GO TO 10
CALL COMSQT(UDISC, VDISC, UDUMMY, VDUMMY)
UBBB=-VA(2)+UDUMMY
0018
0019
0020
0021
0022
0023
                          VBBB=-VA(2)+VDUMMY
                          RDUMMY=~UA(2)-UDUMMY
SDUMMY=~VA(2)-VDUMMY
0024
0025
                          UAAA=2.0*UA(1)
0026
                          VAAA=2.0*VA(1)
0027
0028
                          BBB=UAAA*UAAA+VAAA*VAAA
0029
                          URGOT (NROOT+1) = (UBBB*UAAA+VB6B*VAAA)/B6B
0030
                          VRDDTINRODT+11=[VBBB*UAAA-UBBB*VAAA]/BB8
0031
                          URDOT (NRODT+2) = (ROUMMY+UAAA+SOUMMY+VAAA)/BBB
                          VROOT (NROOT+2) = (SQUMMY*UAAA-ROUMMY*VAAA)/BB8
MULTI(NROOT+1)=1
0032
0033
                          MULTI(NRODT+2)=1
0034
0035
                          NROOT=NROOT+2
                     GO TO 50
10 UAAA=2.0*UA(1)
VAAA=2.0*VA(1)
0036
0037
0036
                          BBB=UAAA*UAAA+VAAA*VAAA
0039
                          URGOT(NROGY+1)=(-UA(2)+UAAA-VA(2)+VAAA)/BBB
0040
0041
                          VROOT (NROOT+1)=(-VA(2)*UAAA+UA(2)*VAAA)/888
0042
                          MULTI(NROOT+1)=2
0043
                          NROOT=NROOT+1
0044
                      50
                         RETURN
0045
                          END
```

TABLE B. VIII (Continued)

```
0001
                            SUBROUTINE COMSQTIUX, VX, UY, VY)
                  00000
                            THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.
                      DOUBLE PRECISION UX, VX, UY, VY, DUMMY, R, AAA, BBB
R=DSQRT(UX+UX+VX+VX)
AAA=DSQRT(DABS({R+UX}/2.0})
BBB=DSQRT(DABS({R+UX}/2.0})
IF(VX) 10,20,30
0002
0003
0004
0006
9007
                      0008
0009
0010
0011
0012
0013
0014
                           UY=0.D
                      Y=DSQRT(DUMMY)
GD TD 100
50 UY=0.0
VY=0.0
0016
0017
0018
0019
                       60 TO 100
0020
0021
                           UY=DSQRT(DUMNY)
0022
                     VY=0.0
100 RETURN
END
0023
0024
0025
```

	,C *	######################################	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
	C *	HORNER'S METHOD COMPUTES THE VALUE OF A POLYNOMIAL P(X) AT A POINT ITS DERIVATIVE AT D. SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE FACTOR (X-D).	F D AND
	C *	POLYNOMIAL BY DIVIDING OUT THE FACTOR (X-0).	
	C *	*******************	******
	_	1)	
002		DOUBLE PRECISION YDPX,RXO,VXO,RB,VB,RC,VC,RPX,VPX,RDPX,RA,VA	
003		DIMENSION RA(26), VA(26), RB(26), VB(26), RC(26), VC(26)	
004		RB(1)=RA(1)	5
005		VB(1)=VA(1)	- 5
006		NUM=NDEF+1	5
07		DO 10 1=2,NUM	5
008	,	RB(1)=RA(1)+(RB(1-1)+RX0-V8(1-1)+VX0)	
009	10	VB(1)=VA([]+(VB(1-1)*RXO+R6(I-1)*VXO)	
10		RPX=RB(NUM)	- 5
11		VPX=VB(NUM)	5
12		RC(1)=RB(1)	5
113		VC(1)=VB(1)	5
014		IF(NDEF.LT.2) GO TO 25	
015		00 20 J=2.NDEF	5
016		RC(J)=RB{J}+{RC(J-1)+RX0-VC(J-1)+VX0}	
017	20	YC(J)=Y8(J)+(YC(J-1)*RXO+RC(J-1)*YXO}	
918	25	RDPX=RC(NDEF)	
019		VDPX=VC(NDEF)	5
20		RETURN	5
021		END	5

0001		SUBROUTINE NEWTON (RPX, VPX, ROPX, VDPX, RXO, VXD, RXNEW, VXNEW)	600
	Ç	**************************************	****
	C	•	*
	C	* THIS SUBROUTINE CALCULATES A NEW APPROXIMATION FROM THE OLD APPROX-	, <b>≉</b>
	C	* [MAT]ON BY USING THE LTERATION FORMULA	•
	C	$* \qquad \qquad K(N+1) = X(N) - P(X(N)) / P^{\bullet}(X(N)).$	*
	С	*	. *
	č	***************************************	****
0002		COUBLE PRECISION RPX.VPX.ROPX.VDPX.RXO.VXO.RXNEW.VXNEW.ARG	
0003		DOUBLE PRECISION DDD	
0004		ARG=RDPX+RDPX+VDPX	
0005		DDD=DSQRT(ARG)	
0006		IF(DDD.EQ.O.O) RETURN	
0007		RXNEW=RXO-((RPX*RDPX+VPX*VDPX)/ARG)	
8000		VXNEW=VXO-((VPX*ROPX-RPX*VOPX)/ARG)	
0009		RETURN	616
0010		END	620

0001	G *	SUBROUTINE CHECK(RPX,VPX,RDPX,VDPX,RXO,VXO,KANS)  ***********************************	***** * * *
	C *	WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.	*
	č *	WILL CO. 13 AS SINCE NO SECURITY SEC	*
	· č *:	*******************	*****
0002		DOUBLE PRECISION RPX,VPX,RDPX,VDPX,RXO,VXO,A85XO,A85QUO,RDUMMY,VOU	749 750
0003		DOUBLE PRECISION ARG	
0804		DOUBLE PRECISION ODD	
0005		COMMON EPS, MAX, ID2	
0006		A8SXC=DSQRT(RXQ#RXC+VXO*VXO)	
0007		IF(ABSXB.EQ.O.) GO TO 25	
8000		ARG=RDPX+RDPX+VDPX+VDPX	
0009		ODD=OSQRT(ARG)	
0010		IF1000.EQ.0.01 G0 T0 25	
0011		RDUMMY=(RPX*RDPX*VPX*VDPX)/ARG VDUMMY=(VPX*RDPX-RPX*VDPX)/ARG	
0012	*	ABSQUD=DSQRT(RDUMMY+RDUMMY+VDUMMY*VDUMMY)	•
0013		IF(ABSQUO/ABSXQ.LT.EPS) GO TO 10	
0014 0015		KANS=0	760
0015		RETURN	764
0017	10	KANS=1	768
0017	10	RETURN	772
0019	25	KANS#0	
0019	2.7	RETURN	
0021		END	780
~~~			

```
000 L
                       SUBROUTINE BETTER(K, RXZERO, VXZERO, RX, VK, NA, RA, VA, RCOEF, VCQEF, N, RC,
                     1VC,RB,V81
                     * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND
                       BY USING THEM AS INITIAL APPROXIMATIONS WITH NEWTON'S METHOD APPLIED TO THE FULL, UNDEFLATED POLYNOMIAL.
                CCC
0002
                       DOUBLE PRECISION RXZERO, VXZERO, RX, VX, RA, VA, RCOEF, VCOEF, RC, VC, RB, V8
                                                                                                                   805
                       1,RXO,YXO,RPX,VPX,RDPX,VDPX,RXNEW,VXNEW,EPS
                      DIMENSION RXZERO(25), VXZERO(25), RX(25), VX(25), RA(26), VA(26), RCOEF(126), VCOEF(26), RC(26), VC(26), RB(26), VB(26)

DOUBLE PRECISION ABPX, ABDPX
0003
0004
0005
                       COMMON EPS, MAX, 102
                       DO 10 (=1,K
RXZERO([)*RX([)
0006
                                                                                                                   612
0007
                    10 VXZERQ([)=VX([]
0008
                                                                                                                    816
                       DO 20 [=1.NA
RCOEF([]=RA([)
0009
0010
                                                                                                                   824
                       VCOEF(II=VALI)
1100
                                                                                                                   825
0012
                       DO 50 J=1,K
                                                                                                                   828
0013
                       RXD=RXZERO(J)
                                                                                                                   832
0014
                       VXO=VXZERO(J)
                                                                                                                   833
                       NN=N
0015
                                                                                                                   834
0016
                       ITER=0
                                                                                                                   836
0017
                   30 CALL HORNER(RCOEF, VCOEF, RXO, VXO, NN, RB, V8, RC, VC, RPX, VPX, RDPX, VDPX)
0018
                       ABPX=DSQRT(RPX*RPX+VPX*VPX)
0019
                       ABDPX=DSQRT(RDPX+RDPX+VDPX+VDPX)
0020
                       IF(ABDPX.NE.0.0) GO TO 33
IF(ABPX.EQ.0.0) GO TO 40
0021
0022
                       GO TO 34
                   33 CALL NEWTON(RPX, VPX, ROPX, VDPX, RXO, VXQ, RXNEW, VXNEW)
0023
0024
                       ITER=ITER+1
                                                                                                                   856
0025
                       RYO=RYNEW
                                                                                                                   860
                       VXO= VX NE W
0026
                                                                                                                   861
                       CALL CHECK(RPX, VPX, RDPX, VDPX, RXC, VXC, KANS)
0027
0028
                       IFIKANS.EQ.11 GO TO 40
                                                                                                                   844
0029
                       IF(ITER.GE.MAX) GO TO 35
                   GO TO 30

34 WRITE(102,1112) RXO,VXO

35 WRITE(102,100) J,RXZERG(J),VXZERG(J)
WRITE(102,200) MAX
0030
                                                                                                                   864
0031
0032
0033
0034
                       RXIJ1=RXO
                                                                                                                   870
0035
                       OXV=(L)XV
                                                                                                                   871
872
0036
                   50 CONTINUE
0037
                       RETURN
                                                                                                                   876
                1112 FORMAT(1H0.36HTHE VALUE OF THE DERIVATIVE AT XO = ,D23.16.3H + ,D2 13.16.2H 1.10H IS ZERO.)
0038
0039
                  100 FORMAT (42HOIN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(.12,4H) = .023
                  1.6.3H + .D23.16.2H 1,18H DIO NOT CONVERGE.)
200 FORMAT(33H THE PRESENT APPROXIMATION AFTER ,13,29H ITERATIONS IS P
0040
                      IRINTED BELOW.
0041
                                                                                                                   880
```

### APPENDIX C

### MULLER'S METHOD

### 2. Use of the Program

A double precision FORTRAN IV program using Muller's method is presented in this appendix. Flow charts for this program are given in Figure C.1 while Table C.V gives a FORTRAN IV listing of this program. Single precision variables are listed in some of the tables. The single precision variables are used in the flow charts and the corresponding double precision variables can be obtained from the appropriate tables.

The program is designed to solve polynomials of degree 25 or less. Both the coefficient of the highest degree term and the constant coefficient should be non-zero. In order to solve polynomials of degree N, where N > 25, certain array dimensions must be changed. These are listed in Table C.I for the main program and subprograms in double precision.

#### TABLE C.I

### PROGRAM CHANGES FOR SOLVING POLYNOMIALS OF DEGREE GREATER THAN 25 BY MULLER'S METHOD

### Double Precision

Main Program

UROOT(N), VROOT(N)
MULT(N)
UAPP(N,3), VAPP(N,3)
UWORK(N+1), VWORK(N+1)
UB(N+1), VB(N+1)
UA(N+1), VA(N+1)
URAPP(N,3), VRAPP(N,3)

Subroutine BETTER

UROOT (N), VROOT (N)
UA (N+1), VA (N+1)
UBAPP (N,3), VBAPP (N,3)
UB (N+1), VB (N+1)
UROOTS (N), VROOTS (N)
URAPP (N,3), VRAPP (N,3)
MULT (N)

Subroutine GENAPP

APPR(N,3), APPI(N,3)

Subroutine HORNER

UA(N+1), VA(N+1) UB(N+1), VB(N+1)

Subroutine QUAD

UA(N+1), VA(N+1) UROOT(N), VROOT(N) MULTI(N)

Table C.II lists the system functions used in the program of Muller's method. In the table "d" denotes a double precision variable name.

TABLE C.II

#### SYSTEM FUNCTIONS USED IN MULLER'S METHOD

#### Double Precision

DABS(d) - obtain absolute value

DATAN2( $d_1, d_2$ ) - arctangent of  $d_1/d_2$ 

DSQRT(d) - square root

DCOS(d) - cosine of angle

DSIN(d) - sine of angle

DSQRT(d) - square root

#### 2. Input Data for Muller's Method

The input data for Muller's method is identical to the input data for Newton's method as described in Appendix B, § 2 except for the variable names. The correspondence of input variable names is given in Table C.III. Only one (not three) initial approximation,  $X_0$ , is given for each root. The other two required by Muller's method are constructed within the program and are  $.9X_0$  and  $1.1X_0$ .

#### 3. Variables Used in Muller's Method

The definitions of the major variables used in Muller's method are given in Table C.IV. For definitions of variables not listed in this table see the definitions of variables for the corresponding subroutine in Table B.VII. The notation and symbols used here are the same as for Table B.VII and are described in Appendix B, § 3.

TABLE C.III

CORRESPONDENCE OF NEWTON'S AND MULLER'S
INPUT DATA VARIABLES

Newton's Method	Muller's Method
Control Card	
NOPOLY N NIAP MAX EPSCNV EPSQ EPSMUL XSTART XEND KCHECK	NOPOLY NP NAPP MAX EPS EPSQ EPSM XSTART XEND KCHECK
Coefficient Car	d
A (RA) A (VA) Initial Approximation	A (UA) A (VA).
	u caru
XZERO (RXZERO) XZERO (VXZERO)	APP (UAPP) APP (VAPP)
End Card	
KCHECK	KCHECK

#### 4. Description of Program Output

The output from Muller's method is the same as that for Newton's method as described in Apptendix B, § 4. Only one initial approximation, Z, (not three) is printed for each root. It is either that supplied by the user or generated by the program. The other two approximations used were 0.9Z and 1.1Z.

#### 5. Informative and Error Messages

The output may contain informative messages printed as an aid to the user. These are:

"NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER XX."

XX is the number of the polynomial. This message is printed if no roots of the polynomial were found.

"IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(X) = YYY

DID NOT CONVERGE AFTER ZZZ ITERATIONS

THE PRESENT APPROXIMATION IS AAA"

X is the number of the root before the attempt to improve accuracy, YYY is the value of the root before attempt to improve accuracy, ZZZ is the maximum number of iterations, and AAA is the current approximation after the maximum number of iterations. This message has the same meaning as the corresponding message in Appendix B, § 5.

TABLE C. IV
VARIABLES USED IN MULLER'S METHOD

Single Precision Variable Type	Double Precision Disposition Variable Type of Argument	Description
	Main	Program
NP I NROOT I NOMULT I ROOT C NAPP I APP C WORK C	NP I NROOT I NOMULT I UROOT, VROOT D NAPP I UAPP, VAPP D UWORK, VWORK D	Degree of polynomial P(X)  Number of distinct roots found  Number of roots (counting multiplicities)  Array containing the roots  Number of initial approximations to be read in  Array of initial approximations  Working array containing coefficients of current  polynomial
B C C C C C	UB, VB D UA, VA D URAPP, VRAPP D	Array containing coefficients of deflated polynomial Array containing coefficients of original polynomial, P(X)
X1 C X2 C X3 C PX1 C	UX1,VX1 D UX2,VX2 D UX3,VX3 D UPX1,VPX1 D	Array of initial or altered approximations for which convergence was obtained  One of three current approximations to a root  One of three current approximations to a root  One of three current approximations to a root  Value of polynomial P(X) at X1
X4 C PX4 C MULT I ITER I I01 I	UPX2, VPX2 D UPX3, VPX3 D UX4, VX4 D UPX4, VPX4 D MULT I ITER I I01 I I02 I	Value of polynomial P(X) at X2  Value of polynomial P(X) at X3  Newest approximation (X <sub>n+1</sub> ) to root  Value of polynomial P(X) at X4  Array containing the multiplicities of each root found  Counter for iterations  Unit number of input device  Unit number of output device
		ANY TOWNER OF ARTHE

Single Precision		cision	Double Pre	cision	Disposition						
<u>Vari</u>	able	Type	Variable	Type	of Argument	<u>Description</u>					
EPSR	T	R	EPSRT	D		Number used in subroutine BETTER to generate two approximations from the one given					
NOPO	LY	I	NOPOLY	I		Number of the polynomial					
MAX		I	MAX	I		Maximum number of iterations					
EPS	-	R	EPS	D		Tolerance check for convergence					
EPS0		R	EPSO	D		Tolerance check for zero (0)					
EP SM		R	EPSM	D		Tolerance check for multiplicities					
KCHE	CK	I	KCHECK	· I		Program control, KCHECK = 1 stops execution of program					
XSTA	RT ·	R	XSTART	D		Magnitude at which to start generating initial approximations					
XEND		R	XEND	D		Magnitude at which to end generating initial approximations					
NWOR	K	I	NWORK	I	·	Degree of current deflated polynomial whose coefficients are in WORK					
ITIM	E	I	ITIME	I		Program control					
NALTI	ER	I	NALTER	Ī	-	Number of alterations which have been performed on an initial approximation					
TAPP		I	IAPP	I		Counter for number of initial approximations used					
CONV		L	CONV	T.		When CONV is true, convergence has been obtained					
IROO'	<b>[</b> '	I	IROOT	Ī		Number of distinct roots solved by Muller's method, i.e. not solved directly by subroutine QUAD					
			· .								
1 .	•				Subro	utine HORNER					
A		С	UA, VA	D	E	Array of current polynomial coefficients (to be deflated or evaluated)					
NA		I	NA	I	E	Degree of polynomial to be deflated or evaluated					
X		С	UX,VX	D	E	Approximation at which to evaluate or deflate the					
					•	polynomial					

TABLE C. IV (Continued)

Single Prec Variable	ision Type	Double Preci Variable		position Argument	<u>Description</u>
В	<b>C</b> ,	UB, VB	D	R .	Array containing the coefficients of the deflated polynomial
PX NUM	C I	UPX,VPX NUM	D I	R	Value of the polynomial at X Number of coefficients of polynomial to be deflated
				Subro	outine TEST
X3 X4	C C	UX3,VX3 UX4,VX4	D	E E	Approximation to Root (old) $(X_n)$ New approximation to root $(X_{n+1})$
CONV EPS EPSO	L R R	CONV EPS EPSO	L.	R C	CONV = true implies convergence has been obtained Tolerance for convergence test
DENOM	R	DENOM	D D		Tolerance check for zero (0)  Magnitude of new approximation, $(X_{n+1})$
				Subrou	itine BETTER
MULT	I	MULT	I	ECR	Array of multiplicities of each root
. A	C	UA, VA	$^{\circ}$ D	E	Array of coefficients of original undeflated polynomial
NP	· I	NP	I	E	Degree of original polynomial
ROOT	C	UROOT, VROOT	D	ECR	Array of roots
NROOT	I	NROOT	I	ECR	Number of roots stored in root
BAPP	С	UBAPP, VBAPP	D	E	Array of initial approximations (old roots)
IROOT	Ι	IROOT	I	ECR	Number of roots solved by the iterative process (Not QUAD)
ROOTS	С	UROOTS, VROOT	S D		Temporary storage for new (better) roots
L	I	L	I.		Number of roots found by BETTER
EPSRT	R	EPSRT	D	, , <b>C</b>	A small number used to generate two of the three
ITER	I	ITER	r	C	approximations when given one Counter for number of iterations

TABLE C. IV (Continued)

Single Precision Variable Type	Double Precision Disposition Variable Type of Argument	Description
B C X1 C X2 C X3 C PX1 C PX2 C PX3 C PX3 C C PX4 C PX4 C X4 C	UB,VB D UX1,VX1 D UX2,VX2 D UX3,VX3 D UPX1,VPX1 D UPX2,VPX2 D UPX3,VPX3 D CONV L UX4,VX4 D	Array containing coefficients of deflated polynomial One of three approximations to the root One of three approximations to the root One of three approximations to the root Value of polynomial (P(X)) at X1 Value of polynomial (P(X)) at X2 Value of polynomial (P(X)) at X3 CONV = true implies convergence has been obtained Newest approximation to root
J	J I	Program control - counts the number of roots used as initial approximations
MAX I IO2 I	MAX I C C I C	Maximum number of iterations permitted Unit number of output device
	Subro	utine ALTER
X1 C X2 C X3 C X2R R X2I R	X1R,X1I D ECR X2R,X2I D ECR X3R,X3I D ECR X2R D X2I D	One of the three approximations to be altered One of the three approximations to be altered One of the three approximations to be altered Real part of complex approximation  Imaginary part of complex approximation
	Subro	utine QUAD
<b>EPST</b> R	EPST D E	Tolerance check for zero (0)
	Subro	utine CALC

These variables are dummy variables used for temporary storage and thus, are not defined.

#### MAIN PROGRAM

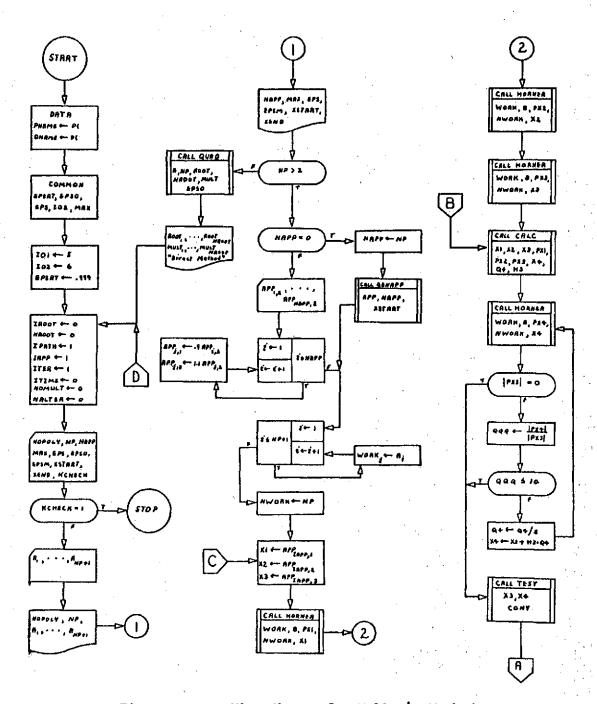


Figure C.1. Flow Charts for Muller's Method

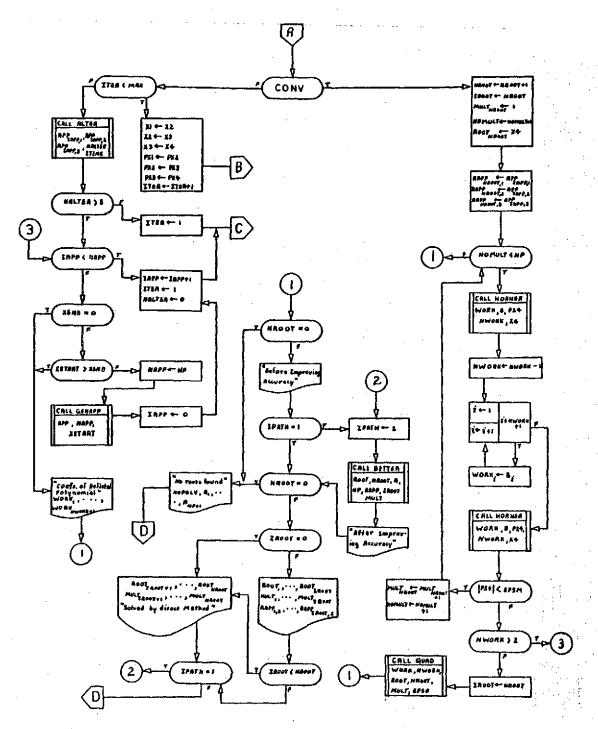


Figure C.1. (Continued)

CALC

TEST

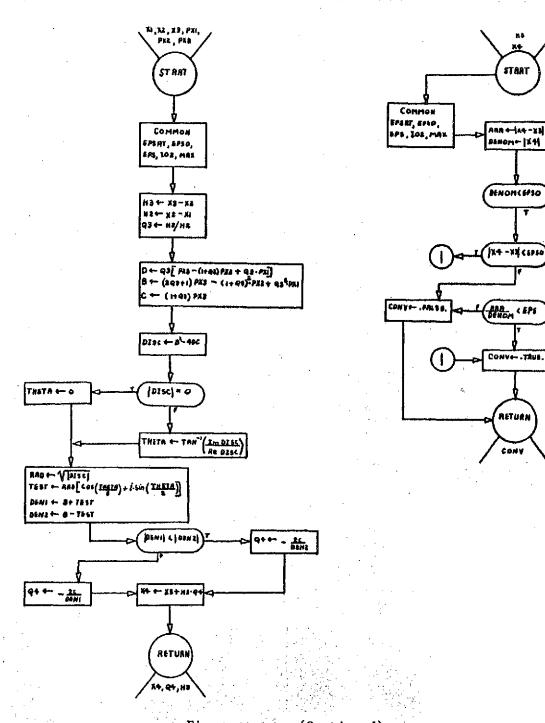


Figure C.1. (Continued)

BETTER

#### HORNER

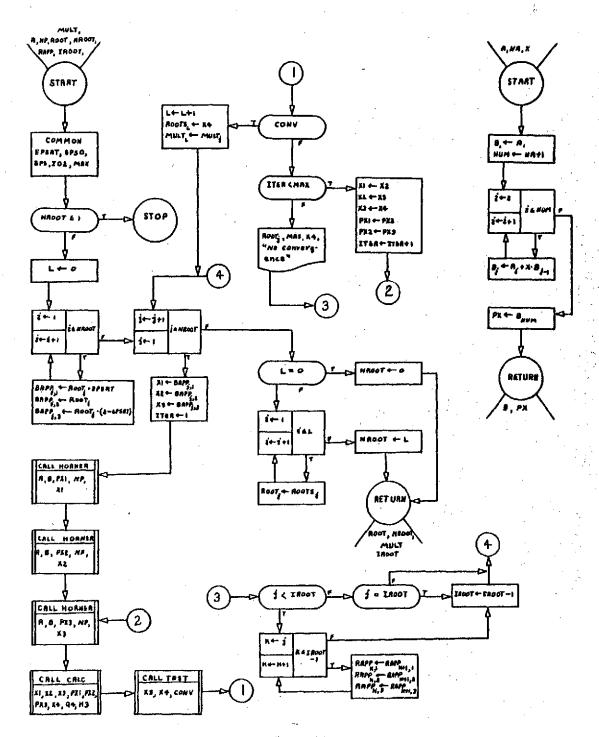


Figure C.1. (Continued)

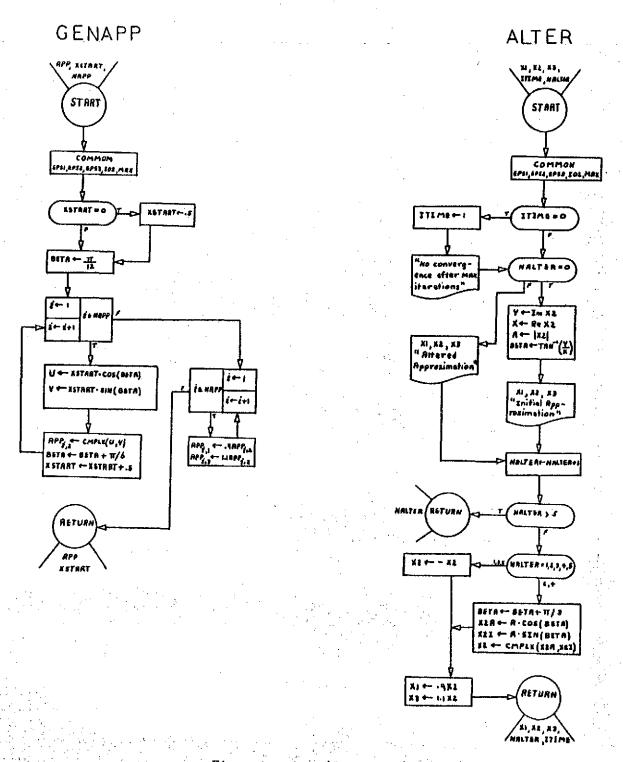
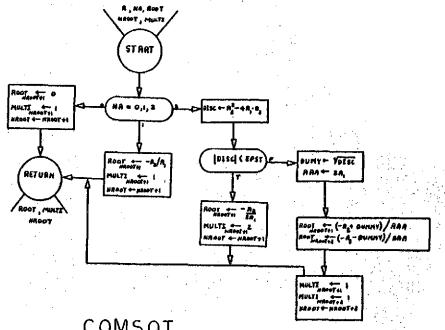


Figure C.1. (Continued)

### QUAD



### COMSQT

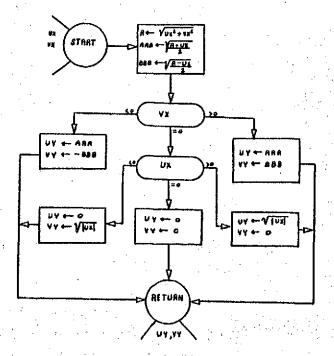


Figure C.1. (Continued)

TABLE C.V.

#### PROGRAM FOR MULLER'S METHOD

```
*******************
                       DOUBLE PRECISION PROGRAM FOR MULLER'S METHOD
                Ċ
                       MULLER'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A POLYNOMIAL OF MAXIMUM DEGREE 25. THROUGH THREE GIVEN POINTS THE POLYNOMIAL IS APPROXIMATED BY A QUADRATIC. THE ZERD OF THE QUADRATIC CLOSEST TO THE OLD APPROXIMATION IS TAKEN AS THE NEW APPROXIMATION.
               C
               С
С
С
                       IN THIS MANNER A SEQUENCE IS OBTAINED CONVERGING TO A ZERO.
                      OOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UROOT,VROOT,UX1,VX1,UAPP,VAPP
1,UX2,VX2,UWORK,VWORK,UX3,VX3,UB,VB,UX4,VX4,UA,VA,UPX1,VPX1,URAPP,V
2RAPP,UPX4,VPX4,EPSRT,EPSO,EPS,CCC,EPSM,UH3,VH3,UQ4,VQ4,ABPX4,ABPX3
0001
                      3,QQQ,XSTART,XEND
                        DIMENSION URODT(25), VROOT(25), MULT(25), UAPP(25,3), VAPP(25,3), UNDRK
0002
                       1(26), VHORK(26), UB(26), VB(26), UA(26), VA(26), URAPP(25,3), VRAPP(25,3)
0003
                       DATA PNAME, DNAME/2HP(,2HD(/
                       LOGICAL CONV
0004
                       COMMON EPSRT, EPSO, EPS, 102, MAX
0005
0006
                        101=5
0007
                        102=6
0008
                        EPSRT=0.999
0009
                    10 NRODT=0
0010
                       IROOT=0
0011
                        IPATH=1
                        NOMULT = 0
0012
                       NAL TER=0
0013
                        1T1ME=0
0014
0015
0016
                        TTER=1
                        READ(101,1000) NOPOLY, NP, NAPP, MAX, EPS, EPSO, EPSM, XSTART, XEND, KCHECK
0017
                        TELECHECK . EO. LI STOP
0018
                        KKK=NP+1
0019
                        READ(101,1010) (UA(1),VA(1),1=1,KKK)
0020
0021
                        WRITE(102,10201 NOPOLY,NP
                        WRITE(102,1035) (PNAME, I, UA(I), VA(I), I=1,KKK)
0022
0023
                        WRITE(102,2060)
                       WRITE(102,2000) NAPP
WRITE(102,2010) MAX
WRITE(102,2020) EPS
0024
0025
0026
                        WRITE(102,2030) EPSM
0027
0028
                        WRITE(102,2040) XSTART
0029
                        WRITE( 102,2050) XEND
                       IF(NP.GT.2) GO TO 15
CALL QUADIUA.VA.NP.URGOT.VROQT.NRGOT.MULT.EPSO)
0030
0031
                        WRITE( 102, 1037)
0032
                        WRITE(102,1086) (1,URDOT(1), VROOT(1), MULT(1), I=1, NROOT)
0033
0034
                       GO TO 10
                    15 IF(NAPP.NE.O) GO TO 20
0035
0036
                       NAPP=NP
                        CALL GENAPP(UAPP, VAPP, NAPP, XSTART)
0037
0038
                        GD TO 27
                    20 READ(101,1030) (UAPPII,2), VAPPII,21,1=1,NAPP)
0039
                       00 25 I=1,NAPP
0040
                        UAPP(1,1)=0.9*UAPP(1,2)
0041
0042
                        VAPP([,1]=0.9*VAPP([,2)
```

```
0043
                          UAPP(1,3)=1.1*UAPP(1,2)
0044
                         VAPP([,3]=1.1*VAPP([,2]
                         KKK=NP+1
DO 30 I=1,KKK
UWORK(I]=UA(I)
0045
0046
0047
0048
                         VWORK(I)=VA(I)
0049
                          NWORK=NP
0050
                         UX1=UAPP([APP.LI
                          VX1=VAPP(IAPP,L)
0051
                          UX2=UAPP([APP,2]
0052
                          VX2=VAPP(JAPP,2)
0053
0054
                          UX3=UAPP([APP,3]
0055
                          VX3=VAPP([APP.3]
                          CALL HORNER(NWORK, UWORK, VWORK, UX1, VX1, UB, VB, UPX1, VPX1)
0056
                     CALL HORNER(NHORK, UHORK, VHORK, UX2, VX2, UB, VB, UPX2, VPX2)

CALL HORNER(NHORK, UHORK, VHORK, UX3, VX3, UB, VB, UPX3, VPX3)

50 CALL CALC(UX1, VX1, UX2, VX2, UX3, VX3, UPX1, VPX1, UPX2, VPX2, UPX3, VPX3, UX

14, VX4, UQ4, VQ4, UH3, VH3;

60 CALL HORNER(NHORK, UHORK, VHORK, UX4, VX4, UB, VB, UPX4, VPX4)
0057
0058
0059
0060
                          ABPX4=OSQRT(UPX4*UPX4+VPX4*VPX4)
0061
0062
                          ABPX3=DSQRT(UPX3+UPX3+VPX3+VPX3)
0063
                          IF(ABPX3.EQ.0.0) GO TO 70
0064
                          QQQ=ABPX4/ABPX3
                          IF('QQQ.LE.10.) GO TO 70
0065
                         UQ4=0.5#UQ4
VQ4=0.5#VQ4
0066
0067
0068
                          UX4=UX3+(UH3+UQ4+VH3+VQ4)
0069
                          VX4=VX3+(VH3+UQ4+UH3+VQ4)
                     VX4=VX3+(VH3*UQ4*UH3*VQ4)
GO TO 60
TO CALL TEST(UX3,VX3,UX4,VX4,CONV)
IF(CONV) GO TO 120
IF(ITER.LT.MAX) GO TO 110
CALL ALTER(UAPP([APP,1],VAPP([APP,1],UAPP([APP,2],VAPP([APP,2],UAP
IP([APP,3],VAPP([APP,3],NALTER,ITIME)
0070
0071
0072
0073
0074
                         IFINALTER GT . 51 GD TO 75
0075
0076
                          ITER=1
                     GO TO 40
75 IF(1APP.LT.NAPP) GO TO 100
1F(XEND.EQ.0.0) GO TO 77
0077
0078
0079
                          IF(XSTART.GT.XEND) GO TO 77
0080
0081
                          CALL GENAPP(UAPP, VAPP, NAPP, XSTART)
0082
0083
                          [APP=0
                     GO TO 100
77 WRITE(102,1090)
0084
0085
                         KKK=NWORK+1
0086
                         WRITE(102,1035) (DNAME, J. UWORK(J), VWORK(J), J=1,KKK)
0087
0088
                     80 IF (NROOT.EQ. 0) GO TO 90
0089
                         WRITE(102,1060)
0090
                         IFTIPATH.EQ.11 GO TO 82
                         IPATH=2
1900
                         CALL BETTER(UA, VA, NP, UROQT, VROQT, NROQT, URAPP, VRAPP, IROQT, MULT)
0092
                          WRITE(102,1200)
0093
                     82 IF(NROOT.EQ.0)GO TO 90
0094
0095
                          IF(IROUT.EQ.O) GO TO 85
0096
                          WRITE(102,1080)
0097
                         DO 55 I=1.1ROOT
                     55 WRITE(102,1085) [,UROOT([),VROOT([),MULT([),URAPP([,2),VRAPP([,2)
0098
```

```
0099
                        IF (IROUT. LT. NROOT) GO TO 85
                    GO TO 87
85 KKK=[ROOT+]
0100
0101
                    WRITE(102,1086) ([,URDOT([],VROOT([],MULT([],[=KKK,NROOT]
87 [F([PATH.EQ.]) GO TO 81
0102
0103
                        GO TO 10
0104
0105
                    90 WRITE(102:1070) NOPOLY
                   GO TO 10
100 [APP=[APP+]
0106
0107
0108
                        ITER=I
                        NALTER=0
GO TO 40
0109
0110
                   120 NROOT=NROOT+1
0111
0112
                         IROOT=NROOT
0113
                        MULT(NROOT)=1
                        NDMULT=NOMULT+1
UROOT(NROOT)=UX4
VROOT(NROOT)=VX4
0114
0115
0116
                        URAPP(NROOT.1)=UAPP(IAPP.1)
                  URAPP(NRUOT,1)=UAPP(IAPP,1)

VRAPP(NROOT,2)=VAPP(IAPP,1)

URAPP(NROOT,2)=VAPP(IAPP,2)

VRAPP(NROOT,2)=VAPP(IAPP,2)

URAPP(NROOT,3)=VAPP(IAPP,3)

VRAPP(NROOT,3)=VAPP(IAPP,3)

125 [FINOMULT.LT.NP] GO TO L30
0118
0120
0121
0122
0123
0124
                        GO TO 80
0125
                        CALL HORNER (NWORK, UWORK, VWORK, UX4, VX4, U8, V8, UPX4, VPX4)
0126
                        NWORK=NWORK-1
0127
                         KKK=NWORK+1
                        00 140 I=1,KKK
UWORK(I)=UB(I)
0128
0129
                   140 VWORK[[]=V8[])
0130
0131
                        CALL HORNER (NWORK, UWORK, VWORK, UX4, VX4, UB, VB, UPX4, VPX4)
0132
                         CCC=DSQRT(UPX4+UPX4+VPX4*VPX4)
                        IF(CCC.LT.EPSM) GO TO 150
IF(NWORK.GT.2) GO TO 75
0133
0134
                        CALL QUADITUMDRK, WHORK, NWORK, URGOT, YRODT, NROOT, MULT, EPSO) GO TO 80
                         1 ROOT = NROOT
0135
0136
0137
0138
                   150 MULT(NROOT) = MULT(NROOT)+1
0139
                        NOMULT=NDMULT+1
                        GO TO 125
0141
                   110 UX1=UX2
                        VX1=VX2
UX2=UX3
0142
0143
                         VX2=VX3
0144
0145
                        UX3=UX4
0146
                         VX3=VX4
0147
                        UPX1=UPX2
0148
                        VPX1≈VPX2
                        UPX2=UPX3
0149
0150
0151
                        VPX2=VPX3
                        UPX3=UPX4
0152
                        VPX3=VPX4
0153
                        ITER= ITER+1
0154
                        GO TO 50
0155
                 1010 FORMAT(2030.0)
                 1020 FORMAT (1HL, 1X, 52 HMULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOM
0156
```

```
11AL/1H .1X,18HPOLYNOMIAL NUMBER ,12.11H OF DEGREE ,12///H ,1X,28H 2THE COEFFICIENTS OF P(X) ARE//)
0157
                                      1030 FORMAT(2030.0)
                                      1090 FORMAT(///,1x,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO 1ZEROS WERE FOUND//)
1080 FORMAT(///1x,13HROOTS OF P(X),52X,14HMULTIPLICITIES,17X,21HINITIAL
0158
0159
                                                 1 APPROXIMATION//)
                                     1 APPROXIMATION///
1070 FORMATI//,43H NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER ,12)
1086 FORMAT(ZX,5HR3OTT,12,4H) = ,D23.16,3H + ,D23.16,2H 1,8X,12,9X,23HS
101VED BY DIRECT METHOD)
1037 FORMAT(//,1X,13HZEROS OF P(XI,51X,14HMULTIPLICITIES//)
0160
2161
                                      1035 FORMAT(3X,42,12,4H) = ,023.16,3H + ,023.16,2H I)
1085 FORMAT(2X,5HROOT(,12,4H) = ,023.16,3H + ,023.16,2H 1,8X,12,8X,D23.
0163
0164
                                    1085 FORMAT(2X,5HROOT(,12,4H) = ,D23.16,3H + ,D23.16,2H 1,8X,12,
116,3H + ,D23.16,2H 1)
1000 FORMAT(3(12,1X),9X,13,8X,3(06.0,1X),13X,2(D7.0,1XI,11)
1060 FORMAT(1//35H BEFORE ATTEMPT TO IMPROVE ACCURACY)
1200 FORMAT(1//1X,37HAFTER THE ATTEMPT TO IMPROVE ACCURACY)
2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN. ,12)
2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,13)
2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,D9.2)
2030 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,D9.2)
2040 FORMAT(1X,21HRADIUS TO END SEARCH.,11X,D9.2)
2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,D9.2)
0165
0166
0167
0168
0169
0170
0171
0172
0173
                                      2060 FORMAT(//1X)
                                                    END
```

```
SUBROUTINE ALTERIXIR, X11, X2R, X21, X3R, X31, NALTER, 171ME)
0001
                  C
C
C
                          SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT.
                  Ċ
                        *
                  C
                           DOUBLE PRECISION XIR,X11,X2R,X21,X3R,X31,EPS1,EPS2,EPS3,R,BETA COMMON EPS1,EPS2,EPS3,102,MAX [F(ITIME.NE.0) GO TO 5
0002
0003
0004
0005
                            [TIME=1
                        WRITE(102,1010) MAX
5 IF(NALTER.EQ.0) GO TO 10
WRITE(102,1000) XLR,XLI,X2R,X2I,X3R,X3I
0006
0007
0008
                           GO TO 20
R=DSQRT(X2R*X2R+X2I*X2I)
0009
0010
0011
                           BETA=DATAN2(X2[,X2R)
                            WRITE(102,1020) X1R,X11,X2R,X21,X3R,X31
0012
                       20 NALTER=NALTER+1

1F(NALTER-GT.5) RETURN

GO TO (30,40,30,40,30),NALTER
0013
0014
0015
                       30 X2R=-X2R
0016
                            X21=-X2[
                           GO TO 50
BETA=BETA+1.0471976
0018
0019
0020
                           X2R=R+DCOS(BETA)
0021
                            X21=R+OSIN(BETA)
                       50 X1R=0.9*X2R
X1I=0.9*X2I
X3R=1.1*X2R
0022
0023
0024
                            X31=1.1*X21
0026
                           RETURN
                    1000 FORMAT(1X,5HX1 = ,D23.16.3H + ,D23.16.2H [,10X,2ZHALTERED APPROXIM 1ATIONS/1X,5HX2 = ,023.16.3H + ,D23.16.2H 1/1X,5HX3 = ,D23.16.3H + 2,D23.16.2H 1/1
0027
                    1020 FORMAT(1H0,5HX1 = ,D23.16,3H + ,D23.16,2H I,1DX,22HINITIAL APPROXI
1MATIONS/tx,5HX2 = ,D23.16,3H + ,D23.16,2H I/1X,5HX3 = ,D23.16,3H +
0028
                          2 ,D23.16,2H 1/1
                    1010 FORMAT(///IX,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
0029
                          ITER ,13,12H ITERATIONS.//)
                           END
 0030
```

```
SUBROUTINE GENAPPLAPPR, APPL, NAPP, XSTART I
0001
            000000
                  SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE DEGREE OF THE ORIGINAL POLYNOMIAL.
                 0002
0003
0004
0005
0006
0007
0008
0009
               BETA=BETA+0.5235988

LO XSTART=XSTART+0.5

DO 20 [=],NAPP

APPR[[,1]=0.9*APPR[[,2]

APPI[[,1]=0.9*APP[[1,2]
0010
0011
0012
0013
0014
                   APPRIL, 31=1.1*APPRIL, 21
0016
                20 APPI(1.31=1.1*APPI(1.2)
0017
                   RETURN
0018
                   END
```

```
"SUBROUTINE BETTERTUA, VA, NP, URODT, TOONN, TOONL, URAPP, TRAPP, I ROOT, MUL
  0001
                       111
                                            _____
                        SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND BY USING THEM AS INITIAL APPROXIMATIONS WITH MULLER'S METHOD APPLIED TO THE FULL, UNDEFLATED POLYNOMIAL.
                 CCC
                        DOUBLE PRECISION URGOT, VROOT, UA, VA, UBAPP, VBAPP, UX1, VX1, UX2, VX2, UX3
0002
                        1,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UB,VB,UROQTS,VRQQTS,EPSRT,UX4,V
                       2X4, URAPP, VRAPP, EPSO, EPS, UQ4, VQ4, UH3, VH3
  0003
                         LOGICAL CONV
                         DIMENSION URDOT(25), VROOT(25), UA(26), VA(26), UBAPP(25,3), VBAPP(25,3
  0004
                       1), UB(26), VB(26), UROOTS(25), VROOTS(25), URAPP(25,3), VRAPP(25,3), MULT
  0005
                        COMMON EPSRT, EPSO, EPS, 102, MAX
  0006
                         [F(NROOT.LE.1) RETURN
  0007
                         L=O
                         DO 10 [=1,NROOT
  0008
                        DO 10 1=1, NKOUT

UBAPP([,1]=UROOT([)*EPSRT

UBAPP([,2]=UROOT([)

VBAPP([,2]=VROOT([)
  0009
  0010
  0011
  0012
                         UBAPP([,3)=URGOT([)*(2.0-EPSRT)
  0013
  0014
                        VBAPP(1,3)=VROOT(1)*(2.0-EPSRT)
                        DO 100 J=1,NROGT
UX1=UBAPP(J,1)
  0015
  0016
                        VX1=VBAPP(J,1)
  0017
                         UX2=UBAPP(J,2)
  0018
                         VX2=VBAPP(J,2)
  0019
                         UX3=UBAPP(J,31
  0020
  0021
                         VX3=VBAPP(J.3)
  0022
                         [TER=1
                    0023
  0024
  0025
  0026
                        14, VX4, UQ4, VQ4, UH3, VH3}
                    14, VX4, U04, VQ4, UH3, VH3)
30 CALL TEST(UX3, VX3, UX4, VX4, CONV)
IF(CONV) GO TO 50
IF(ITER.LT.MAX) GO TO 40
WRITE(102, 1000) J, UROOT(J), VROOT(J), MAX
HRITE(102, 1010) UX4, VX4
IF(J.LT.IROOT) GO TO 33
  0027
  0028
  0029
  0030
  0031
  0032
                         IF(J.EQ. IROUT) GO TO 35
  0033
  0034
                         GO TO 100
                     33 KKK#IROOT-1
  0035
                        DO 34 K=J.KKK
URAPP(K.1)=URAPP(K+1.1)
  0036
  0037
                         VRAPP(K.11=VRAPP(K+1.1)
  0038
                         URAPP(K,21=URAPP(K+1,2)
  0039
                         VRAPPIK, 2)=VRAPP(K+L, 2)
  0040
                         URAPP(K.31=URAPP(K+1.31
  0041
                         VRAPP(K+31=VRAPP(K+1+3)
  0042
                        IRODI= IROOT-1
  0043
                         GO FO 100
  0044
                     40 UX1=UX2
  0045
```

```
0046
0047
                             VX1≃VX2
UX2=UX3
                            VX2=VX3
UX3=UX4
VX3=VX4
0048
0049
0050
0051
                             UPX1=UPX2
0052
                             VPX1=VPX2
                            UPX2=UPX3
VPX2=VPX3
1TER=ITER+1
GD TO 20
0053
0054
0055
0056
0057
                       50 L=L+1
URODTS(L)=UX4
0058
0059
                             VROOTS(L)=VX4
                      MULT(L)=MULT(J)

100 CONTINUE

1F(L.EQ.0) GO TO 120

DO 110 [#1,L

URODT(I)=UROOTS(I)
0060
0061
0062
0063
0064
0065
                      110 VROOT(11=VROOTS(1)
0066
                             NROOT=L
0067
                             RETURN
0068
0069
                      120 NROOT=0
                    RETURN

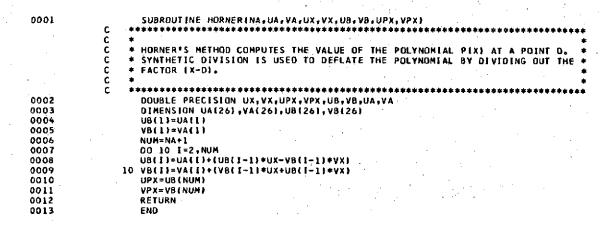
1000 FORMAT(///42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,12,4H) = ,
1023.16,3H + .023.16,2H I/24H DID NOT CONVERGE AFTER ,13,11H ITERAT
210NS1
0070
0071
                     1010 FORMAT(30H THE PRESENT APPROXIMATION IS .D23.16.3H + .D23.16.2H I/
                           1/)
END
0072
```

```
SUBROUTINE CALCIUXI, VXI, UX2, VX2, UX3, VX3, UPX1, VPX1, UPX2, VPX2, UPX3, V
0001
                     ¢
                       GIVEN THREE APPROXIMATIONS x(n-2), x(n-1), and x(n), subroutine calcapproximates the polynomial by a quadratic and solves for the zero of the quadratic closest to x(n). This zero is the new approximation x(n+1) to the zero of the polynomial.
                200
                Ċ
                                                  C
                      DOUBLE PRECISION ARG1.ARG2
DOUBLE PRECISION UPX3.VPX3.UPX2.VPX2.UX1.VX1.UX2.VX2.UX3.VX3.UPX1.
IVPX1.UH3.VH3.UH2.VH2.UQ3.VQ3.UD.VD.UB.VB.UC.VC.UDISC.VDISC.UCCC.VC
2CC.UDENI.VDENI.UDEN2.VDEN2.UQ4.VQ4.UX4.VX4.EPSRT.EPSO.EPS.UDDD.VDD
3D.AAA.BBB.RAD.UAAA.VAAA.UBBB.VBB
DOUBLE PRECISION THETA.ANGLE.UTEST.VTEST
COMMON EPSRT.EPSO.EPS.IO2.MAX
                        DOUBLE PRECISION ARGI.ARG2
0002
0003
0004
0005
                        UH3=UX3−UX2
0006
0007
                        VH3=VX3-VX2
                        UH2=UX2-UX1
0008
0009
                        VH2=VX2-VX1
                        BBB=UH2+UH2+VH2+VH2
0010
                        UQ3=(UH3+UH2+VH3+VH2)/888
0011
                        VQ3=[VH3+UH2-UH3+VH2]/B8B
0012
                        UDDD=1.0+U03
0013
                        VDDD=VQ3
0014
                        UD=(UPX3-(UDDD+UPX2-YDDD+VPX2))+(UQ3+UPX1-YQ3+VPX1)
0015
                        VD=[VPX3-(VDDD+UPX2+UDDD+VPX2)]+[VQ3+UPX1+UQ3*VPX1)
0016
0017
                        UAAA=2.0+UQ3
                        VAAA=2.0*VQ3
UAAA=UAAA+1.0
0018
0019
                        UBBB=UDOD*UDDO-VDDO*VDDD
0020
                        V888=VD0D*UDDD+UDDD*V0DD
0021
                        UCCC=UQ3+UQ3-VQ3+VQ3
0022
                        VCCC=VQ3+UQ3+UQ3+VQ3
0023
                        UB=((UAAA*UPX3-VAAA*VPX3)-(UBBB*UPX2-VBBB*VPX2))+(UCCC*UPX1-VCCC*V
0024
                       1PX11
                        Y8={(YAAA*UPX3+UAAA*VPX3)~(Y888*UPX2+U888*YPX2)]+(YCCC*UPX1+UCCC*Y
0025
                       1 P X 1 1
                        UC=UDDD+UPX3-VDDD+VPX3
0026
                        VC=VDDD+UPX3+U00D+VPX3
0027
                        UDISC={UB+UB-YB*VB}-{4.0*(UD*UC-YD*VC)}
VDISC={2.0*(VB*UB)}-(4.0*(VD*UC+UD*VC)}
0028
0029
                        AAA=DSQRT(UDISC*UDISC+VDISC*VDISC)
0030
                        IF(AAA.EQ.0.0) GO TO 5
0031
                        GO TO 7
0032
                        THETA=0.0
0033
0034
                        GO TO 9
                        THETA=DATAN2(VDISC.UDISC)
0035
                        RAD=DSQRT(AAA)
ANGLE=THETA/2.0
0036
0037
                        UTEST=RAD+DCOS (ANGLE)
0038
                         VTEST=RAD+DS IN (ANGLE)
0039
                        UDEN1≈UB+UTEST
0040
0041
                         VDENL=VB+VTEST
0042
                        UNEN2=UA-UTEST
                        VDEN2= VB-VTEST
0043
                        ARGI=UDEN1+UDEN1+VDEN1+VDEN1
```

0044

0046 0047 0048 0049 1F(AAA.LT.BBB) GD TO 10 0049 1F(AAA.EQ.O.O) GD TO 60 0050 0050 004AA=-2.0*UC 0051 VAAA=-2.0*UC 0052 UQ4=(UAAA*UDEN1+VAAA*VDEN1)/ARG1 0053 0055 10 1F(BBB.EQ.O.O) GO TO 60 0055 10 1F(BBB.EQ.O.O) GO TO 60 0056 004A=-2.0*UC 0057 VAAA=-2.0*UC 0057 VAAA=-2.0*UC 0058 004=(UAAA*UDEN2+VAAA*VDEN2)/ARG2 0059 VQ4=(UAAA*UDEN2+VAAA*VDEN2)/ARG2 0059 0060 0061 50 UX4=-UX3+(UH3*UQ4-VH3*VQ4) 0062 VX4=VX3+(VH3*UQ4+UH3*VQ4) 0064 0004 0065 0066 GD TO 50 0066 GD TO 50 0066 GD TO 50	0045	*ARG2=UDEN2*UDEN2+VDEN2*VDEN2
0048	0046	AAA=DSQRT (ARGL)
0049	0047	BBB=DSQRT(ARG2)
0050	0048	IFIAAA.LY.BBB) GD TO 10
0050 0051 VAAA=-2.0*UC 0051 VAAA=-2.0*UC 0052 UQ4=(UAAA*UDENI+VAAA*VDENI)/ARG1 0053 VQ4=(VAAA*UDENI-UAAA*VDENI)/ARG1 0054 GD TO 50 0055 10 If(BBB.EQ.O.O) GO TO 60 0056 VAAA=-2.0*UC 0057 VAAA=-2.0*UC 0058 UQ4=(UAAA*UDEN2+VAAA*VDEN2)/ARG2 0059 VQ4=(VAAA*UDEN2-UAAA*VDEN2)/ARG2 0060 GD TO 50 0061 50 UX4=UX3+(UH3*UQ4-VH3*VQ4) 0062 VX4=VX3+(VH3*UQ4+UH3*VQ4) 0063 RETURN 0064 60 UQ4=1.0 0065 VQ4=0.0	0049	IFLAAA.EO.O.O. GD TO 60
0051		UAAA=-2.0≠UC
0052		VAAA=-2.0+VC
0053		
0054 0055 10 If(888.EQ.Q.O) GO TO 60 0056 0057 VAAA=-2.0*VC 0058 0059 0059 0060 00 GO TO 50 0061 50 UX4=UX3+(UH3*UQ4-YH3*VQ4) 0062 0063 0064 00 QO TO 50 0064 00 QO TO 50 0065 0065 0065 0065 0066 0070 0066		
0055 10 IF(8BB.EQ.Q.O) GO TO 60 0056		
0056		
0057		
0058		+·····
0059		
0060 GD TO 50 0061 50 UX4=UX3+(UH3*UQ4-VH3*VQ4) 0062 VX4=VX3+(VH3*UQ4+UH3*VQ4) 0063 RETURN 0064 60 UQ4*1.0 0065 VQ4=0.0 0066 GD TO 50		
0061 50 UX4=UX3+(UH3*UQ4-VH3*VQ4) 0062		
0062		
0063 RETURN 0064 60 UQ4*1.0 0065 VQ4=0.0 0066 GD TO 50		
0064 60 UQ4=1.0 0065 VQ4=0.0 0066 GD TO 50		the state of the s
0065 V04=0.0 0066 GD TO 50		
0066 GD TD 50		
		• •
0067 END	0066	
	0067	END

```
0001
                              SUBROUTINE TESTILIX3, VX3, UX4, VX4, CONVI
                    00000
                             SUBROUTINE TEST CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-
IMATIONS BY TESTING THE EXPRESSION
ABSOLUTE VALUE OF (X(N+1)-X(N))/ABSOLUTE VALUE OF X(N+1).
WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.
                              DOUBLE PRECISION UX3,VX3,UX4,VX4.EPSRT.EPSO.EPS.AAA.UDUMMY,VDUMMY,
0002
                             1DENOM
                             LOGICAL CONV
COMMON EPSRT, EPSO, EPS, 102, MAX
0003
0004
0005
                              UDUMMY=UX4-UX3
                              VDUMMY=VX4-VX3
0006
0007
                              AAA=DSQRT(UDUMMY*UDUMMY*VDUMMY*VDUMMY)
0008
                              DENOM=DSQRT(UX4+UX4+VX4+VX4)
0009
0010
0011
0012
                         IF(DENOM.LT.EPSO) GO TO 20
IF(AAA/DENDM.LT.EPS) GO TO 10
5 CONV=.FALSE.
GO TO 100
10 CONV=.TRUE.
0013
0014
                              GO TO 100
0015
                             IF(AAA.LT.EPSO) GO TO 10
                             GO TO 5
RETURN
0016
0017
                       100
0018
                              END
```



```
SUBROUTINE QUADIDA, VA. NA. UROOT, VROOT, NROOT, MULTI, EPST)
0001
                   C,
                             SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR. SOLUTION OF THE QUADRATIC IS DONE USING THE QUADRATIC FORMULA.
                   000
                          ٠
                   C
                                                      **************
                           DOUBLE PRECISION UA, VA, UROOT, VROOT, 888, UAAA, VAAA, UDISC, VDISC, UDUMM LY, VDUMMY, ROUMMY, SDUMMY, EPST, UBBB, VBBB DIMENSION UA(26), VA(26), UROOT(25), VROOT(25), MULTI(25) [F(NA.EQ.2] GD TO 7 [F(NA.EQ.1] GD TO 5
0002
0003
0004
0005
                             URDOT(NROOT+1)=0.0
0006
                             VROOT (NROOT+1)=0.0
0007
                             MULTI(NROOT+1)=1
0008
0009
                             NRDOT=NROOT+1
0010
                             GO TO 50
                          5 888=UA(11*UA(1)*VA(1)*VA(1)
UROOT(NROOT+1)=(-UA(2)*UA(1)-VA(2)*VA(1))/888
VROOT(NROOT+1)=(-VA(2)*UA(1)+UA(2)*VA(1))/888
0011
0012
0013
                             MULTI(NROOT+1)=1
0014
0015
                             MRODT=NROOT+1
                         NRODT=NRUUT+1
GO 10 50
7 UDISC=(UA(2)*UA(2)*VA(2)*VA(2))-(4.0*(UA(1)*UA(3)-VA(1)*VA(3)))
VDISC=(VA(2)*UA(2)*UA(2)*VA(2)]-(4.0*(VA(1)*UA(3)*UA(1)*VA(3)))
RBB=DSQRT(UDISC*UDISC*VDISC*VDISC)
IF(BBB,LT.EPST) GO 10 10
CALL COMSQT(UDISC,VDISC,UDUMHY,VDUMHY)
0016
0017
0018
0019
0020
0021
                             UBBB=-UA(2)+UDUMMY
0022
                             VBBB=-VA(2)+VDUMMY
0023
                             ROUMMY=-UA(2)-UDUMMY
SOUMMY=-VA(2)-VOUMMY
0024
0025
                             UAAA=2.0*UA(1)
VAAA=2.0*VA(1)
0026
0027
                             BBB=UAAA+UAAA+VAAA*VAAA
0028
                             UROOT(NROOT+1)=(U8B8*UAAA+V8B8*YAAA)/BBB
0029
                             VROOT(NROOT+1)={VBBB*UAAA-UBBB*VAAA}/BBB
UROOT(NROOT+2)={ROUMMY*UAAA+SOUMMY*VAAA}/BBB
0030
0031
                             VROOT (NROOT+2)=(SOUMMY*UAAA-ROUMMY*VAAA)/BBB
MUL TI (NROOT+1)=1
0032
0033
                             MULTI(NRODT+2)=1
0034
                             NRUOT=NROOT+2
0035
                             GO TO 50
0036
                        10 UAAA=2.0*UA(1)
0037
                             VAAA=2.0*VA(1)
888=UAAA*UAAA+VAAA*VAAA
0038
0039
                             URDUT(NROOT+1)=(-UA(2)*UAAA-VA(2)*VAAA)/888
0040
                             VROOT (NROOT+1) = (-VA(2) +UAAA+UA(2) +VAAA) / BBB
0041
                             MULTI(NROUT+1)=2
0042
                             NROOT=NROOT+1
0043
                         50 RETURN
0044
                             END
```

```
SUBROUTINE COMSQT(UX, VX, UY, VY)
0001
                   00000
                             THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.
                            DOUBLE PRECISION UX, VX, UY, VY, DUMMY, R, AAA, 888
R=DSQRT(UX=UX+VX=VX)
AAA=DSQRT(DA8S1(R+UX1/2.0))
0002
0003
0004
0005
                             BBB=DSQRT(DABS((R-UX)/2.01)
0006
                             [f(VX) 10,20,30
0007
                        10 UY=AAA
                        VY=-1.0+BBB
GG TO 100
20 IF(UX1 40,50,60
30 UY=AAA
8000
0009
0010
0011
0012
0013
                             VY=B8B
                        GO TO 100
40 DUMMY=DABS(UX)
0014
                        UY=0.0
VY=DSQRT(DUMMY)
GO TO 100
50 UY=0.0
0015
0016
0017
0018
0019
                             VY=0.0
                        GO TO 100
60 DUMMY=DABS(UX)
UY=DSQRT(DUMMY)
VY=0.0
0020
0021
0022
0023
                      100 RETURN
0024
                             END
0025
```

#### APPENDIX D

## SPECIAL FEATURES OF THE G.C.D. AND THE REPEATED G.C.D. PROGRAMS

Several special features have been provided in each program as an aid to the user and to improve accuracy of the results. These are explained and illustrated below.\*

#### 1. Generating Approximations

If the user does not have initial approximations available, sub-routine GENAPP can systematically generate, for an N<sup>th</sup> degree polynomial, N initial approximations of increasing magnitude, beginning with the magnitude specified by XSTART. If XSTART is 0., XSTART is automatically initialized to 0.5 to avoid the approximation 0. + 0.i. The approximations are generated according to the formula:

$$X_{K} = (XSTART + 0.5K) (Cos \beta + i Sin \beta)$$

where

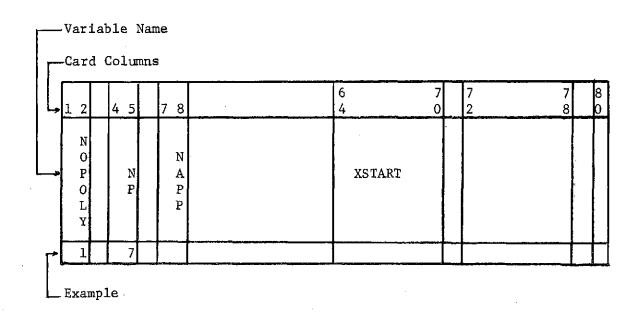
$$\beta = \frac{\Pi}{12} + K \frac{\Pi}{6}$$
,  $K = 0,1,2,...$ 

To accomplish this, the user defines the number of initial approximations to be read (NAPP) on the control card to be zero (0) or these columns

<sup>\*</sup>These illustrations are representative of G.C.D.-Newton's method in double precision. Control cards for other methods should be prepared accordingly.

(7-8) may be left blank. If XSTART is left blank, it is interpreted as 0.

For example, a portion of a control card which generates initial approximations beginning at the origin for a seventh degree polynomial is shown in Example D.1.



Example D.1

The approximations are generated in a spiral configuration as illustrated in Figure A.1.

Example D.2 shows a portion of a control card which generates initial approximations beginning at a magnitude of 25.0 for a sixth degree polynomial.

1	2	4 5	7 8	6 7 4 0	7 2 `	7 8	8 0
	N O P O L	N P	N A P P	XSTART			
	2	6		2.5D+01			$oxed{\Box}$

Example D.2

Note that if the approximations are generated beginning at the origin, the order in which the roots are found will probably be of increasing magnitude. Roots obtained in this way are usually more accurate.

#### 2. Altering Approximations

If an initial approximation, X<sub>0</sub>, does not produce convergence to a root within the maximum number of iterations, it is systematically altered a maximum of five times until convergence is possibly obtained according to the following formulas:

If the number of the alteration is odd: (j = 1,3)

$$X_{i+1} = |X_0|$$
 (Cos  $\beta + i$  Sin  $\beta$ ) where

$$\beta = \text{Tan}^{-1}$$
  $\frac{\text{Im } X_0}{\text{Re } X_0} + K \frac{\text{II}}{3}$ ;  $K = 1 \text{ if } j = 1$   
 $K = 2 \text{ if } j = 3$ .

If the number of the alteration is even: (j = 0,2,4)

$$x_{j+1} = -x_{j}$$
.

and the second of the second o

Each altered approximation is then taken as a starting approximation. If none of the six starting approximations produce convergence, the next initial approximation is taken, and the process repeated. The six approximations are spaced 60 degrees apart on a circle of radius  $|X_0|$  centered at the origin as illustrated in Figure A.2.

#### 3. Searching the Complex Plane

By use of initial approximations and the altering technique, any region of the complex plane in the form of an annulus centered at the origin can be searched for roots. This procedure can be accomplished in two ways.

The first way is more versatile but requires more effort on the part of the user. Specifically selected initial approximation can be used to define particular regions to be searched. For example, if the roots of a particular polynomial are known to have magnitudes between 20 and 40 an annulus of inner radius 20 and outer radius 40 could be searched by using the initial approximations  $20 \cdot + 1$ ,  $23 \cdot + 1$ ,  $26 \cdot + 1$ ,  $29 \cdot + 1$ ,  $32 \cdot + 1$ ,  $35 \cdot + 1$ ,  $38 \cdot + 1$ ,  $40 \cdot + 1$ .

By generating initial approximations internally, the program can search an annulus centered at the origin of inner radius XSTART and outer radius XEND. Values for XSTART and XEND are supplied on the control card by the user. Example D.3 shows a portion of a control card to search the above annulus of inner radius 20.0 and outer radius 40.0.

1	2	4	5	7 8	6 7 4 0	7 7 2 8	8 0
	N O P O L		N P	N A P	XSTART	XEND	
	I		7		2.0D+01	4.0D+01	

Example D.3

Note that since not less than N initial approximations can be generated at one time, the outer radius of the annulus actually searched may be greater than XEND but not greater than XEND  $\pm$  .5N.

Example D.4 shows a control card to search a circle of radius 15.

1	2	45	7 8	6 7 4 0	7 7 2 8	8 0
	N O P O L Y	N P	N A P P	XSTART	XEND	
	1	7			1.5D+01	

Example D.4

Figure A.3 shows the distribution of initial and altered approximations for an annulus of width 2 and inner radius a.

#### 4. Improving Zeros Found

After the zeros of a polynomial are found, they are printed under the heading "Roots of Q(X)." They are then used as initial approximations with Newton's (Muller's) method applied each time to the full (undeflated) polynomial Q(X), which contains only distinct roots. In most cases, zeros that have lost accuracy due to roundoff error in the deflation process are improved. The improved zeros are then printed under the heading "Roots of P(X)." Since each root is used as an approximation to the original (undeflated) polynomial Q(X), it is possible that the root may converge to an entirely different root. This is especially true where several zeros are close together. Therefore, the user should check both lists of zeros to determine whether or not this has occurred.

#### 5. Solving Quadratic Polynomial

After N-2 roots of an N<sup>th</sup> degree polynomial have been extracted, the remaining quadratic,  $ax^2 + bx + c$ , is solved using the quadratic formula

$$X = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

for the two remaining roots. These are indicated by the words "Results of Subroutine QUAD" in the initial approximation column. If only a polynomial of degree I is to be solved, the solution is found directly as (X - C) = 0 implies X = C.

#### 6. Missing Roots

If not all N roots of an N<sup>th</sup> degree polynomial are found, the coefficients of the remaining deflated polynomial are printed under the heading "Coefficients of Deflated Polynomial For Which No Zeros Were Found." The user may then work with this polynomial in an attempt to find the remaining roots. The leading coefficient (coefficient of the highest degree term) will be printed first (Exhibit 6.11)

#### 7. Miscellaneous

By using various combinations of values for NAPP, XSTART, and XEND, the user has several options available as illustrated below.

Example D.5 shows the control card for a seventh degree polynomial. Three initial approximations are supplied by the user. At most three roots will be found and the coefficients of the remaining deflated polynomial will be printed.

1 2	45	~ 7 8	6 7 4 0	7 7 2 8	8 0
N O P C L Y	N P	N A P P	XSTART	XEND	
1	7	3			

Example D.5

Note that if several roots are known to the user, they may be "divided out" of the original polynomial by using this procedure.

Example D.6 indicates that 2 initial approximations are supplied by the user to a  $7^{th}$  degree polynomial. After these approximations are used the circle of radius 15 will be searched for the remaining roots.

1	2	4 5	7 8	6 7 4 0	7 7 2 8	8 0
	N O P O L Y	N P	N A P P	XSTART	XEND	
	1	7	2		1.5D+01	

Example D.6

By defining XSTART between 0. and 15. an annulus instead of the circle will be searched.

Solver for the first the second terror of the secon

#### APPENDIX E

#### G.C.D. - NEWTON'S METHOD

#### 1. Use of the Program

A double precision FORTRAN IV program using the G.C.D. method with Newton's method as a supporting method is presented here. Flow charts for this program are given in Figure E.6 while Table E.VII gives a FORTRAN IV listing of this program. Single precision variables are listed in some of the tables. The simple precision variables are used in the flow charts and the corresponding double precision variables can be obtained from the appropriate tables.

This program is designed to solve polynomials having degree less than or equal to 25. In order to solve polynomials of degree N where N > 25, the data statement and array dimensions given in Talbe E.I must be changed.

In this program both the leading coefficient and the constant coefficient are assumed to be non-zero.

#### TABLE E.I.

# PROGRAM CHANGES NECESSARY TO SOLVE POLYNOMIALS OF DEGREE GREATER THAN 25 BY G.C.D. - NEWTON'S METHOD

#### Main Program

Subroutine GENAPP

APPR(N), APPI(N)

Subroutine GCD

UR(N+1), VR(N+1) US(N+1), VS(N+1) USS(N+1), VSS(N+1) URR(N+1), VRR(N+1) UT(N+1), VT(N+1)

Subroutine QUAD

UA(N+1), VA(N+1) UROOT(N), VROOT(N) MULT(N)

Subroutine NEWTON

UP(N+1), VP(N+1) UB(N+1), VB(N+1)

Subroutine DIVIDE

UP(N+1), VP(N+1) UD(N+1), VD(N+1) UQ(N+1), VQ(N+1)

Subroutine HORNER

UP(N+1), VP(N+1) UB(N+1), VB(N+1)

Subroutine DERIV

UP(N+1), VP(N+1) UA(N+1), VA(N+1)

Subroutine MULTI

UP(N+1), VP(N+1)
UROOT(N), VROOT(N)
UA(N+1), VA(N+1)
UB(N+1), VB(N+1)
MULT(N)

## 2. Input Data for G.C.D. - Newton's Method

The input data for G.C.D. - Newton's method is grouped into polynomial data sets. Each polynomial data set consists of the data for one and only one polynomial. As many polynomials as the user desires may be solved by placing the polynomial data sets one behind the other. Each polynomial data set consists of three kinds of information placed in the following order:

- 1. Control information.
- 2. Coefficients of the polynomial.
- Initial approximations. These may be omitted as described in Appendix D, § 1.

An end card follows the entire collection of data sets. It indicates that there is no more data to follow and terminates execution of the

program. This information is displayed in Figure E.1 and described below. All data should be right justified and the D-type specification should be used. The recommendations given in Table E.II are those found to give best results on the IBM 360/50 computer which has a 32 bit word.

## Control Information

The control card is the first card of the polynomial data set and contains the information given in Table E.II. See Figure E.2.

TABLE E.II

CONTROL DATA FOR G.C.D. - NEWTON'S METHOD

Variable Name	Card Columns	Description
NOPOLY	c.c. 1-2	Number of the polynomial. Integer. Right justified.
NP	c.c. 4-5	Degree of the polynomial. Integer. Right justified.
NAPP	c.c. 7-8	Number of initial approximations to be read. Integer. Right justified. If no initial approximations are given, leave blank.
MAX	c.c. 19-21	Maximum number of iterations. Integer. Right justified. 200 is recommended.
EPS1	c.c. 23-28	Test for zero in subroutine GCD. Double precision. Right justify. 1.D-03 is recommended.

TABLE E.II (Continued)

Variable Name	Card Columns	Description
EPS2	c.c. 30-35	Convergence requirement. Double precision. Right justify. 1.D-10 is recommended.
EPS3	c.c. 37-42	Test for zero in subroutine QUAD. Double precision. Right justify. 1.D-20 is recommended.
EPS4	c.c. 44-49	Multiplicity requirement. Double precision. Right justify. 1.D-02 is recommended.
XSTART	c.c. 64-70	Magnitude at which to begin generating initial approximations. Double precision. Right justify. This is a special feature of the program and may be omitted.
XEND	c.c. 72-78	Magnitude at which to end the generating of initial approximations.  Double precision.  Right justify.  This is a special feature of the program and may be omitted.
KCHECK	c.c. 80	This should be left blank.

## Coefficients of the Polynomial

The coefficient cards follow the control card. For an N<sup>th</sup> degree polynomial, N+1 coefficients must be entered one per card. The coefficient of the highest degree term is entered first; that is, the leading coefficient is entered first. For example, if the polynomial  $x^5 + 3x^4 + 2x + 5$  were to be solved for its zeros, the order in which

the coefficients would be entered is: 1, 3, 0, 0, 2, 5. Each real or complex coefficient is entered, one per card, as described in Table E.III and illustrated in Figure E.3.

TABLE E.III

COEFFICIENT DATA FOR G.C.D. - NEWTON'S METHOD

Variable Name	Card Columns	Description
UP (P in single precision)	c.c. 1-30	Real part of complex coefficient. Double precision. Right justify. If none, leave blank or enter 0.0000.
VP (P in single precision)	c.c. 31-60	Imaginary part of complex coefficient. Double precision. Right justify. If none, leave blank or enter 0.0D00.

## Initial Approximations

The initial approximation cards follow the set of coefficient cards. The number of initial approximations read must be the number specified on the control card and are entered, one per card, as given in Table E.IV and illustrated in Figure E.4.

TABLE E.IV

INITIAL APPROXIMATION DATA FOR G.C.D. - NEWTON'S METHOD

Variable Name	Card Columns	Description
UAPP (APP in single precision)	c.c. 1~30	Real part of complex number. Double precision. Right justify. If none, leave blank or enter 0.0000.
VAPP (APP in single precision)	c.c. 31-60	Imaginary part of complex number. Double precision. Right justify. If none, leave blank or enter 0.0000.

## End Card

The end card is the last card of the input data to the program.

It indicates that there is no more data to be read. When this card is read, program execution is terminated. This card is described in Table E.V and illustrated in Figure E.5.

TABLE E.V

DATA TO END EXECUTION OF G.C.D. - NEWTON'S METHOD

Variable Name	Card Column	Description
KCHECK	c.c. 80	Must contain the number 1. Integer.

## 3. Variables Used in G.C.D. - Newton's Method

The definitions of the major variables used in G.C.D. - Newton's method are given in Table E.VI. The symbols used to indicate type are:

R - real variable

I - integer variable

D - double precision

C - complex variable

L - logical variable

A - alphanumeric variable

When two variables are listed, the one on the left is the real part of the corresponding single precision complex variable; the one on the right is the imaginary part. The symbols used to indicate disposition are:

E - entered

R - returned

ECR - entered, changed, and returned

C - variable in common

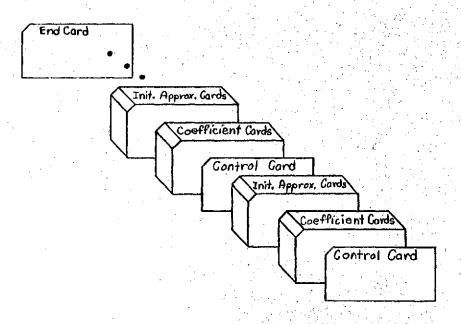


Figure E.1. Sequence of Input Data for G.C.D.-Newton's Method

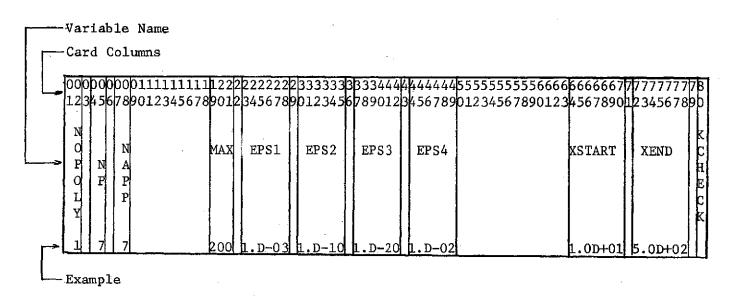


Figure E.2. Control Card for G.C.D. - Newton's Method

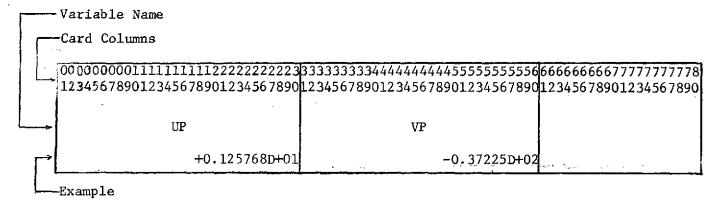


Figure E.3. Coefficient Card for G.C.D. - Newton's Method

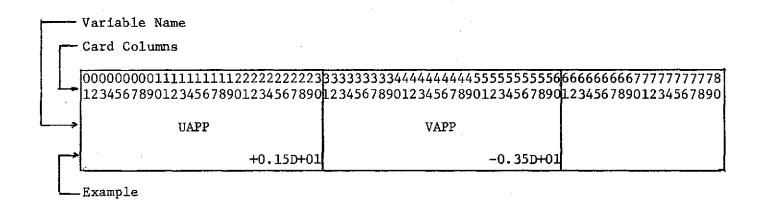


Figure E.4. Initial Approximation Card for G.C.D. - Newton's Method

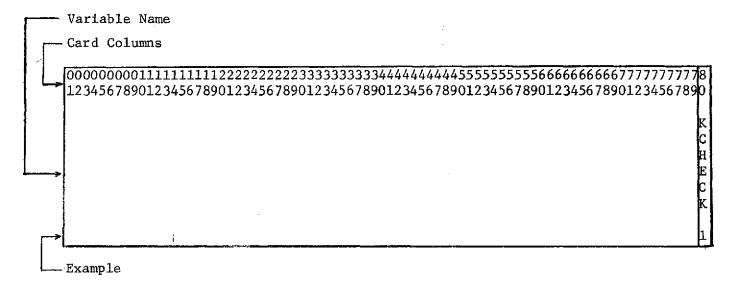


Figure E.5. End Card for G.C.D. - Newton's Method

TABLE E.VI

VARIABLES USED IN G.C.D. - NEWTON'S METHOD

Single Pre Variable	cision Type	<u>Double Prec:</u> <u>Variable</u>	ision Type	Disposition of Argument	Description
				Main	Program
J	I	J	I	]	Number of distinct roots found
ITIME	I	ITIME	I	Ì	Program control
NOPOLY	I	NOPOLY	I	1	Number of the polynomial
NP	1	NP	I		Degree of the original polynomial
P	С	UP,VP	D		Array of coefficients of original polynomial, P(X)
NAPP	I	NAPP	I		Number of initial approximation to be read
EPS1	R	EPS1	D	7	Tolerance check for zero (0) in Subroutine GCD
EPS2	R	EPS2	D		Tolerance check for convergence
EPS3	Ŕ	EPS3	D	•	Tolerance check for zero (0) in Subroutine QUAD
MAX	I	MAX	I	1	Maximum number of iterations permitted
101	I	I01	I	1	Unit number of input device
102	I	102	I	1	Unit number of output device
KCHECK	I	KCHECK	I	]	Program control, KCHECK = 1 implies stop execution
APP	C	UAPP, VAPP	D		Array of initial approximations
XSTART	R	XSTART	D	I	Magnitude at which to start search for roots
XEND	R	XEND	D	1	Magnitude at which to end search for roots
ANAME	Α	ANAME	A	(	Contains name of method used "NEWTONS"
ROOT	С	UROOT, VROOT	D		Array of roots found
MULT	I	MULT	I	I	Array of multiplicities
DP	С	UDP, VDP	D		Array containing coefficients of the derivative, $(P'(X))$ , of $P(X)$
NDP	I	NDP	I	I	Degree of the derivative of original polynomial
D	С	UD, VD	D		Array of coefficients of the greatest common divisor of P(X) and P'(X)
ND	I	ND	I	Î	Degree of g.c.d. of P(X) and P'(X)
Q	С	UQ,VQ	D		Array of coefficients of quotient polynomial P(X)/g.c.d.

TABLE E.VI (Continued)

Single Pre	cision	Double Precision		Disposition	•
Variable	Type	Variable	Type	of Argument	Description
NQ	I	NQ	I		Promo of muchdook action of 1 0/20
ZRO	C	UZRO,VZRO	D T		Degree of quotient polynomial Q(X)
DUMMY	c	UDUMMY VDUMM			Value at which to evaluate or deflate polynomial
QQ	C	•			Dummy variable
NQQ NQQ	I	UQQ,VQQ	D		Working array of coefficients of current polynomial
IALTER	· I	NQQ	I.		Degree of current polynomial, QQ(X)
		IALTER	I.		Number of alterations of an initial approximation
CONV	L	CONV	L		CONV = TRUE implies convergence to a root
EPS4	R	EPS4	D		Tolerance for checking multiplicities
AP	С	UAP, VAP	D		Array of approximations (initial or altered) producing convergence
QD	С	UQD, VQD	D		Array of coefficients of newly deflated polynomial
JAP	I	JAP	1		Number of distinct roots found by iterative process
		•			i.e. not as a result of Subroutine QUAD
Jl	I	J1	I		Number of distinct roots found in the attempt to improve
					roots
ROOTS	С	UROOTS VROOTS	D		Array of improved roots
NEWT	L	NEWI	L.		Program control. NEWT = TRUE implies that Newton's
					method was used instead of Subroutine QUAD
				C. T	Add NITHON
				Supro	outine NEWTON
X	C	UX,VX	D	${f E}$	Starting approximation (initial or altered)
N	I	N	I	$\mathbf{E}$	Degree of current polynomial
P	С	UP,VP	D	E	Array of coefficients of current polynomial
MAX	I	MAX	I	С	Maximum number of iterations
EPSLON	R	EPSLON	D	С	Tolerance for checking convergence
X0	C	UXO,VXO	D	R	Current approximation to root
В	С	UB, ÝB	D		Array of coefficients of newly deflated polynomial
DPXO	Ċ	UDPXO VDPXO	D		Derivative of the polynomial at XO
	-	, ,			was partiament or 200

TABLE E.VI (Continued)

Single Pre Variable	cision Type	Double Preci Variable	sion Type	Disposition of Argument	Description
	<u> </u>		<u> </u>	<u> </u>	
DIFF	С	UDIFF, VDIFF	D		PXO/DPXO
PX0	С	-	D		Value of polynomial at XO
CONV	L	CONV	L	R	CONV = TRUE implies convergence to root
				Subro	outine HORNER
X	С		D	E	Value at which to evaluate or deflate polynomial
N	I	N	I	E	Degree of polynomial
P	С	UP, VP	D		Array of coefficients of polynomial
C	С	UC,VC	D	R	Updated at each iteration to yield derivative of poly-
					nomial at X
В	С	UB, VB	D		Array of coefficients of newly deflated polynomial
			 	Sub	routine QUAD
N	I	N	I	E	Degree of polynomial to be solved
A	С	UA,VA	. D	E	Array of coefficients of polynomial to be solved
J	Ι	J	I	ECR	Number of distinct roots found of original polynomial $(J = -1 \text{ implies original polynomial is of degree 2 or 1})$
ROOT	С	UROOT, VROOT	D	ECR	Array of roots found
MULT	I	MULT	I	ECR	Array of multiplicities
DISC	C	UDISC, VDISC	D		Discriminate of quadratic
TEMP	С	UTEMP, VTEMP	D		√DISC
EPSLON	R	EPSLON	D	С	Tolerance for zero (0)
D	С	UD, VD	D		Twice leading coefficient of quadratic

TABLE E.VI (Continued)

Single Pre Variable	cision Type	Double Preci Variable	Ision Type	Disposition of Argument	Description
				Sub	routine GCD
R	С	UR, VR	D	E	Array of coefficients of original polynomial, P(X)
S	Ç	US,VS	D	E	Array of coefficients of derivative polynomial, P (X)
N	I	N	I	E	Degree of original polynomial, P(X)
M	I	M	I	E	Degree of derivative polynomial, P'(X)
RR	C	URR, VRR	D		Array of coefficients of dividend polynomial
SS	C	uss, vss	D	R	Array of coefficients of divisor polynomial also array
					of coefficients of $g.c.d.$ of $P(X)$ and $P'(X)$ when
					returned
N1	I	N1	I		Degree of dividend polynomial, RR(X)
M1	I	M1	I	R	Degree of divisor polynomial, SS(X), also degree of
					g.c.d. of $P(X)$ and $P'(X)$ when returned
D	С	UD, VD	D		Quotient RR <sub>N1+1</sub> /SS <sub>M1+1</sub>
${f T}$	C	UT,VT	D		Array of coefficients of difference polynomial(RR - D(SS))
K	I	K ·	I		Degree of difference polynomial T(X)
EPSLON	R	EPSLON	D	С	Tolerance check for zero (0)
				Subro	outine MULTI
N	I	N	I	E	Degree of original polynomial, P(X)
P	С	UP,VP	D	E	Array of coefficients of original polynomial, P(X)
J	I	J	I	E	Number of distinct roots of P(X)
ROOT	C	UROOT, VROOT	D	E	Array of distinct roots of P(X)
A	C	UA,VA	D		Working array of coefficients of current polynomial
М	I	M	I		Degree of current polynomial, A(X)
$\mathtt{MULT}$	I	MULT	I	R	Array of multiplicities of the roots
102	I	102	I	C	Unit number of output device
В	С	UB, VB	D		Array of coefficients of newly deflated polynomial
С	С	UC, VC	D		Derivative of polynomial at ROOT <sub>i</sub>
EPSLON	R	EPSLON	D	С	Tolerance for checking multiplicities

TABLE E.VI (Continued)

Single Pre Variable	Type	Doubel Prec Variable	ision Type	Disposition of Argument	Description
				Subro	outine DERIV
N	I	N	I	<b>E</b> ·	Degree of polynomial, P(X)
P	C	UP,VP	D	E	Array of coefficients of polynomial, P(X)
A	С	UA,VA	D	R	Array of coefficients of derivative, P'(X)
М	I	M	I	R	Degree of derivative polynomial, P'(X)
				Subro	outine DIVIDE
P	С	UP, VP	D	E	Array of coefficients of dividend polynomial
N	I	N	I	E	Degree of dividend polynomial
D	C	UD, VD	D	E	Array of coefficients of divisor polynomial
M	1	М .	I	E	Degree of divisor polynomial
Q	C	UQ,VQ	D	R	Array of coefficients of quotient polynomial $P(X)/D(X)$
K	1	K	I	R	Degree of quotient polynomial, Q(X)
J	I.	J	1		Counter
TERM	С	UTERM, VTERM	D	•	Dummy variable used for temporary storage of products
KK	I	KK	I		Number of coefficients of quotient polynomial, Q(X)
				Subro	outine GENAPP
APP	C	APPR, APPI	D	R	Array containing initial approximations
NAPP	I	NAPP	I	${f E}$ .	Number of initial approximations to be generated
XSTART	R	XSTART	D	ECR	Magnitude at which to begin generating approximations; also magnitude of the approximation being generated
BETA	R	BETA	D		Argument of complex approximation being generated
U	R	APPR(I)	D		Real part of complex approximation
V	R	APPI(I)	D		Imaginary part of complex approximation



Single Pre Variable	cision Type	Double Preci Variable	sion Type	Disposition of Argument	Description
				Subro	outine ALTER
XOLD	С	XOLDR, XOLDI	D	ECR	Old approximation to be altered to new approximation
NALTER	I	NALTER	I	ECR	Number of alterations performed on an initial approximation
ITIME	I	ITIME	I	${f E}$	Program control
MAX	I	MAX	I	Ç	Maximum number of iterations permitted
Y	R	XOLDI	D		Imaginary part of original initial approximation (unaltered)
X	R	XOLDR	D		Real part of original, unaltered initial approximation
R	R	ABXOLD	D		Magnitude of original unaltered initial approximation
BETA	R	BETA	D		Argument of new approximation
XOLDR	R	XOLDR	D		Real part of new approximation
XOLDI	R	XOLDI	D		Imaginary part of new approximation
102	I	102	I	С	Unit number of output device
				Subr	coutine COMSQT
		ux,vx	D	E	Complex number for which the square root is desired
		UY,VY	D	R	Square root of the complex number

### 4. Description of Program Output

The output from G.C.D. - Newton's method consists of the following information.

The heading is "GREATEST COMMON DIVISOR METHOD USED WITH NEWTON'S METHOD TO FIND ZEROS OF POLYNOMIALS NUMBER XX." XX represents the number of the polynomial.

As an aid to ensure that the control information is correct, the number of initial approximations given, maximum number of iterations, test for zero in subroutine GCD, test for convergence, test for zero in subroutine QUAD, test for multiplicities, radius to start search, and radius to end search are printed as read from the control card.

The coefficients of the polynomial are printed under the heading "THE DEGREE OF P(X) IS XX THE COEFFICIENTS ARE." XX represents the degree of the polynomial. The coefficient of the highest degree term is printed first.

The polynomial obtained after dividing the original polynomial, P(X), by the greatest common divisor of P(X) and its derivative, P'(X), is printed under the heading "Q(X) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE DISTINCT ROOTS OF P(X). THE DEGREE OF Q(X) IS XX THE COEFFICIENTS ARE." XX represents the degree of this polynomial. This polynomial contains all distinct roots and is solved by Newton's method. The coefficient of the highest degree term is printed first; that is, the leading coefficient is printed first.

The zeros found before the attempt to improve accuracy are printed under the heading "ROOTS OF Q(X)."

The initial approximation producing convergence to a root is

APPROXIMATION." The initial approximations may be those supplied by the user, or generated by the program or a combination of both. The message "RESULTS OF SUBROUTINE QUAD" indicates that the corresponding root was obtained by subroutine QUAD. See Appendix D, § 5.

The zeros found after the attempt to improve accuracy are printed under the heading "ROOTS OF P(X)." The corresponding initial approximation producing convergence is printed as described above.

The multiplicity of each zero is given under the title "MULTIPLICITIES."

## 5. Informative Messages and Error Messages

The output may contain informative or error messages. These are intended as an aid to the user and are described as follows.

If not all roots of a polynomial were found before the attempt to improve accuracy, the remaining unsolved polynomial will be printed, with the leading coefficient first, under the heading "COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND." See Appendix D, § 6.

"NO ROOTS FOR INITIAL APPROXIMATION ROOT XX = YYY." This message is printed if a root fails to produce convergence when trying to improve accuracy. XX represents the number of the root and YYY represents the value of the root before the attempt to improve accuracy.

"NO ROOTS FOR THE POLYNOMIAL Q(X) OF DEGREE XX WITH GENERATED

INITIAL APPROXIMATIONS." XX represents the degree of the polynomial

Q(X). This message is printed if none of the roots produce convergence in the attempt to improve accuracy.

"THE EPSILON (XXX) CHECK IN SUBROUTINE MULTI INDICATES THAT ROOT

YY = ZZZ IS NOT CLOSE ENOUGH TO BE A TRUE ROOT. IT IS PRINTED BELOW

WITH MULTIPLICITY O." XXX represents the multiplicity requirement

(EPS4 on the control card), YY represents the number of the root, and

ZZZ represents the value of the root after the attempt to improve

accuracy. The message indicates that this root does not meet the

requirement for multiplicities. It is, however, usually a good

approximation to the true root since convergence was obtained both

before and after the attempt to improve accuracy.

## MAIN PROGRAM

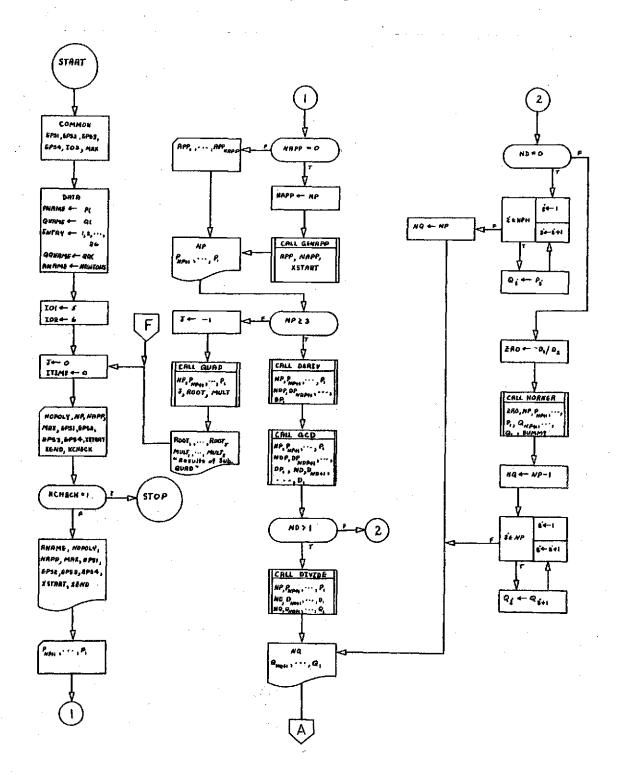


Figure E.6. Flow Charts for G.C.D.-Newton's Method

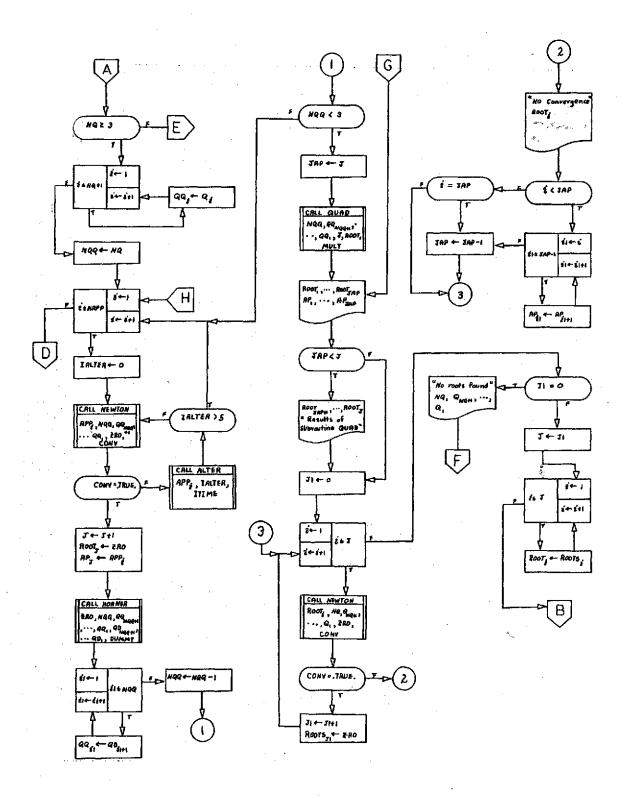


Figure E.6. (Continued)

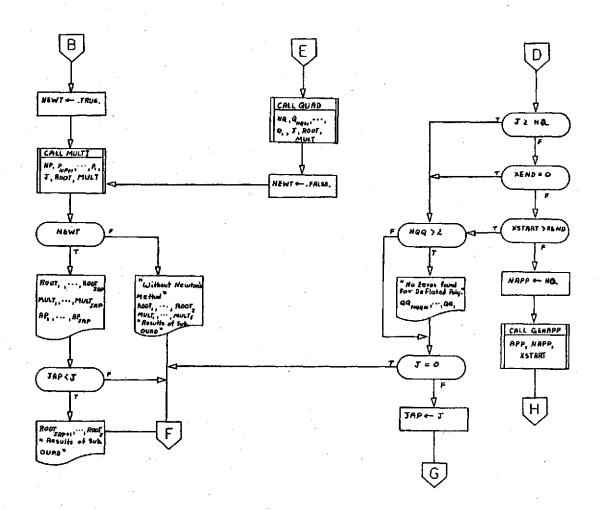


Figure E.6. (Continued)

QUAD

NEWTON

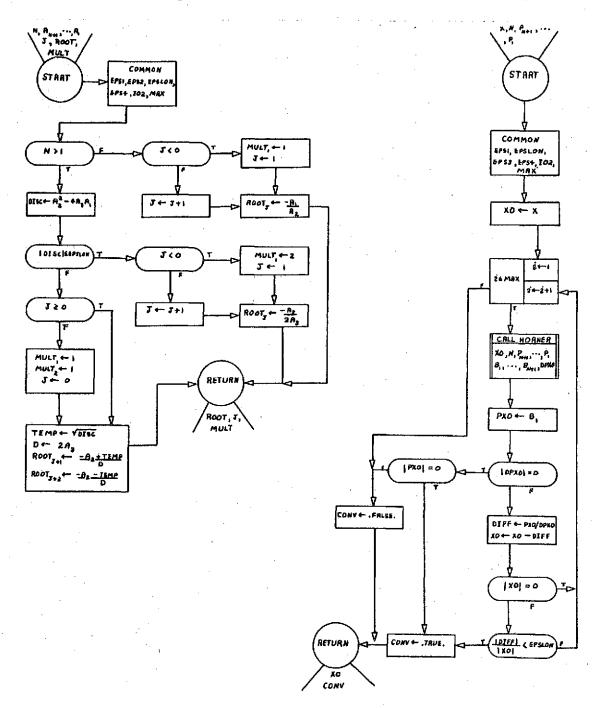


Figure E.6. (Continued)

# HORNER DIVIDE START START $\begin{array}{c} B_{N+1} \leftarrow P_{N+1} \\ B_{N} \leftarrow XB_{N+1} + P_{N} \\ C \leftarrow B_{N+1} \end{array}$ X+- H-M Q+- PHI/DMI ž<del>4-</del>1 RETURN B. -×8,,, + 8,-1 c ← x·c + β BETURN J← J+1 TERM -- P DERIV START KK>I MIZI RETURN A. -- (1-1) P. KH ← KK -1 Q - TERM DMI

Figure E.6. (Continued)

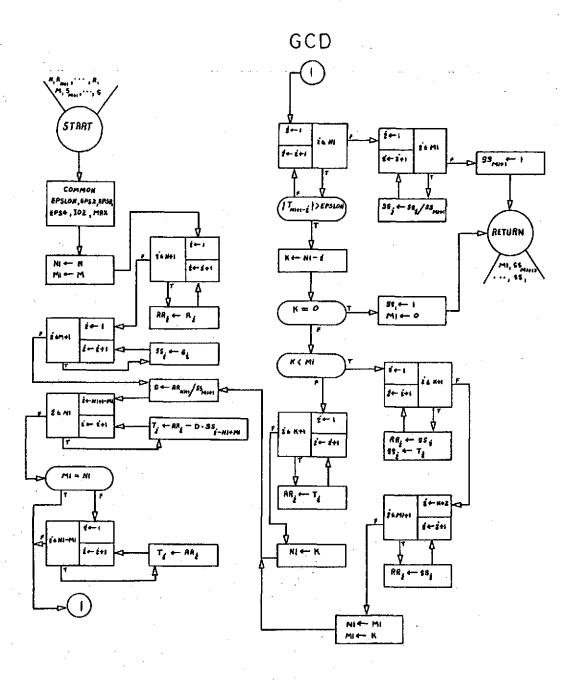


Figure E.6. (Continued)

# MULTI M, PHEL ..., P. ... STRRT COMMON EPSI, EPS2, EPS3, EPSLON, TO2, MAX M← N CALL HORNER ROOT, M, A, A, B, , MULT, =0

Figure E.6. (Continued)

# COMSQT

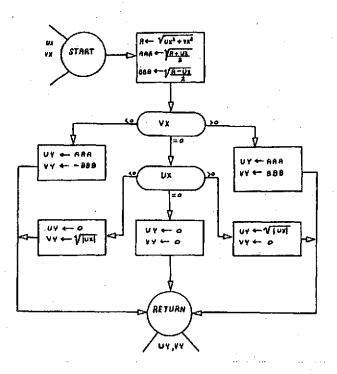
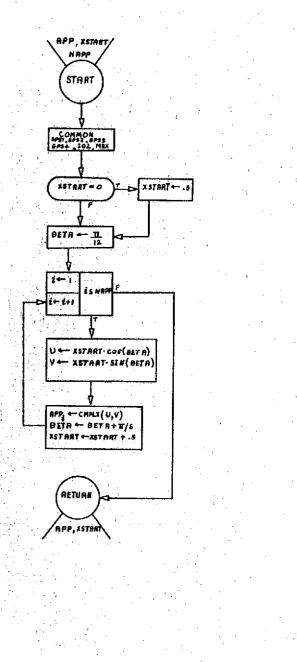


Figure E.6. (Continued)

# GENAPP

# **FLTER**



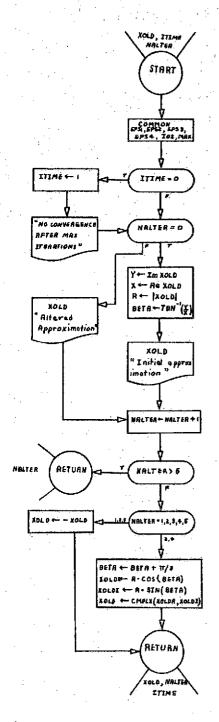


Figure E.6. (Continued)

#### TABLE E.VII

#### PROGRAM FOR G.C.D.-NEWTON'S METHOD

```
*************
              C
C
                     DOUBLE PRECISION PROGRAM FOR G.C.D. - NEWTON'S METHOD
              Ç
                   * THE G.C.O. METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A
                   * POLYNOMIAL OF MAXIMUM DEGREE 25. ALL MULTIPLE ROOTS ARE REMOVED BY *
* DIVIDING THE POLYNOMIAL BY THE GREATEST COMMON DIVISOR OF THE POLYNOMIAL *
* AND ITS DERIVATIVE. THE ZEROS OF THE RESULTING POLYNOMIAL ARE EXTRACTED *
* AND THEIR MULTIPLICITIES DETERMINED. *
              C
              0
0001
                     DOUBLE PRECISION UP, VP, UAPP, VAPP, UROOT, VROOT, UDP, VDP, UD, VD, UZRO, VZ
                     1RO,UQ,VQ,UDUMMY,VDUMMY,UQQ,VQQ,UAP,VAP,UQD,VQD,UROQTS,VRQQTS,EPS1,
                    2EPS2.EPS3.EPS4
0002
                     DOUBLE PRECISION XSTART
0003
                     DOUBLE PRECISION XEND
0004
                     DIMENSION UP(26), VP(26), UAPP(25), VAPP(25), UROOT(25), VROOT(25), MULT
                     1(25),UDP(26),VDP(26),UD(26),VD(26),UQ(26),VQ(26),UQQ(26),UQQ(26),U
                    2AP[25], VAP(25), UQD(26), VQD(26), ANAME(2), ENTRY(26), UROOTS(25), VROOT
                    35(25)
0005
                     COMMON EPS1, EPS2, EPS3, EPS4, IO2, MAX
0006
                     LOGICAL NEWT, CONV
0007
                     DATA PNAME.QNAME.QQNAME/2HP(,2HQ(,3HQQ(/
0008
                     DATA ENTRY/1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,2H10,2H11,ZH12,2H13
                    1,2H14,2H15,2H16,2H17,2H18,2H19,2H20,2H21,2H22,2H23,2H24,2H25,2H26/
0009
                     DATA ANAME(1), ANAME(2)/4HNEWT, 4HONS /
0010
                     1.01 = 5
0011
                     102=6
0012
                  10 J=0
0013
                      O=3M1TI
0014
                     READ(101.1000) NOPOLY.NP.NAPP.MAX.EPS1.EPS2.EPS3.EPS4.XSTART.XEND.
                    1KCHECK
0015
                     1F(KCHECK.EQ.1) STOP
                     WRITE[102,1020] ANAME(1), ANAME(21, NOPOLY
6100
0017
                     WRITEI102,20001 NAPP
0018
                     WRITE(102,2010) MAX
0019
                     WRITE(102,2070) EPS1
0020
                     WRITE(102,2020) EPS2
0021
                     WRITE(102-2080) EPS3
                     WRITE(102,2030) EPS4
WRITE(102,2040) XSTART
0022
0023
                     WRITE(102,2050) XEND
0024
0025
                     WRITE( 102, 2060)
                     KKK=NP+1
0026
0027
                     NNN=KKK+1
0028
                     DO 20 I=1.KKK
                     I-MNN=LLC
0029
                 20 READ(101,1010) UP(JJJ), VP(JJJ)
IF(NAPP, NE.O) GO TO 22
0030
0031
                     NAPP=NP
0032
0033
                     CALL GENAPP (UAPP, VAPP, NAPP, XSTART)
0034
                     GO TO 23
                 22 READ([01,1015] (UAPP(I), VAPP(II, I=1, NAPP)
0035
0036
                 23 WRITE(102,1030) NP
                     KKK=NP+L
0037
                     NNN=KKK+1
0039
                     DO 25 1=1.KKK
0039
```

```
0040
                           I-MMM=EEE
0041
                          WRITE([02,1040) PNAME, ENTRY(JJJ), UP(JJJ), VP(JJJ)
0042
                           IF(NP.GE.3) GO TO 30
0043
                           J=- 1
                           CALL QUAD(NP,UP,VP,J,URDOT,VROOT,MULT)
WRITE([02,1070]
WRITE([02,1165] ([,UROOT([),VROOT([),MULT([],[=1,J]).
0044
0045
0046
                      GO TO 10
30 CALL DERIVINE, UP, VP, NOP, UOP, VDP)
0047
0048
                          CALL GCD(NP, UP, VP, NDP, UDP, VDP, ND, UD, VD)
IF(ND.GT.1) GD TO 70
IF(ND.EQ.0) GD TO 65
0049
0050
0051
                          UDUMMY={UD(2)*UD(2)}+(VD(2)*VD(2)}
UZRO=(-{UD(1)*UD(2)}-{VD(1)*VD(2)}/UDUMMY
VZRO=(-{UD(2)*VD(1)}+{UD(1)*VD(2)}/UDUMMY
0052
0053
0054
                           CALL HORNER (UZRO, VZRO, NP, UP, VP, UQ, VQ, UDUMMY, VDUMMY)
0055
0056
                           NQ=NP-L
                      00 60 [=1,NP

UQ([)=UQ([+1)

60 VQ([)=VQ([+1)

G0 TG 80
0057
0058
0059
0060
0061
                          KKK=NP+1
                          DO 66 [=].KKK
0062
0063
                          (1)9V=(1)DV
0064
                          NQ=NP
0065
0066
                          GO TO 80
                      TO CALL DIVIDE(NP.UP.VP.ND.UD.VD.NQ.UQ.VQ)
80 WRITE(102.1120) NQ
0067
0068
                          KKK=NQ+1
0069
                          NNN=KKK+1
0070
0071
                          DO 83 1=1,KKK
0072
                           1 -- NNN=ししし
                      83 WRITE(102,1040) QNAME, ENTRY(JJJ), UQ(JJJ), VQ(JJJ) IF(NQ.GE.3) GO TO 85 GO TO 110
0073
0074
0075
0076
                      85 KKK=NQ+1
                      00 90 I=1+KKK
000(I)=U0(I)
0077
0078
0079
0080
                          NGO=NQ
1800
                          GO TO 120
                    110 CALL QUAD(NQ,UQ,VQ,J,UROOT,VROOT,MULT)
NEWT=.FALSE.
GO TO 310
120 DO 200 I=1,NAPP
0082
0083
0084
0085
                          IALTER=0
0086
                    130 CALL NEWTON (UAPP(II, VAPP(I), NQQ, UQQ, VQQ, UZRO, VZRO, CONVI
0087
                          IFICONY) GO TO 160
CALL ALTER(UAPP(I), VAPP(I), IALTER, ITIME)
IFIIALTER.GT.5) GO TO 200
0088
CORG
0090
                          GO TO 130
0091
                          J=J+L
UROOT(J)=UZRO
VRDOT(J)=VZRO
0092
0093
0094
0095
                          UAP(J)=UAPP(I)
0096
                          CALL HORNER (UZRO, VZRO, NQQ, UQQ, VQQ, UQD, VQD, UDUMMY, VDUMMY)
0097
```

```
0098
                          DO 180 | 1=1, NQQ
                          U901111=U9D111+11
0099
                    180 VQQ(11)=VQQ(11+1)
NQQ=NQQ-1
0100
0101
                           IF(NOQ.LT.3) GO TO 220
0102
0103
                    200 CONTINUE
                          IF(J.GE.NQ) GO TO 205
IF(XEND.EQ.O.D) GO TO 205
IF(XSTART.GT.XEND) GO TO 205
0104
0105
0106
                          NAPP=NQ
0107
0108
                          CALL GENAPP (UAPP, VAPP, NAPP, XSTART)
                    GO TO 120
205 IF(NQQ-LE-2) GO TO 210
0109
0110
                          WRITE(102+1200)
0111
0112
                          KKK=NOQ+1
                          NNN=KKK+1
0113
                          DD 157 L=1.KKK
0114
                          1-NNN=L
                    157 WRITE(102,1100) QQNAME,ENTRY[JJJ],UQQ(JJJ),VQQ(JJJ)
210 IF(J.EQ.0) GO TO 10
JAP=J
0116
0117
0118
                          GO TO 230
0119
                    220 JAP=J
0120
                    CALL QUADINQQ,UQQ,VQQ,J,UROOT,VROOT,MULT)

230 WRITE(IO2,1132)

WRITE(IO2,1133) (1,UROOT(I),VROOT(I),UAP(I),VAP(I),I=1,JAP)

IF(JAP,LT,J) GO TO 235

GO TO 240
0121
0122
0123
0124
0125
                    235 KKK=JAP+1
0126
0127
                          WRITE([02,1134] ([,UROOT([),VROOT([],[=KKK,J)
0128
                    240 J1=0
                          DO 300 [=1,J
CALL NEWTON(UROOT(1),VROOT([],NQ,UQ,VQ,UZRO,VZRO,CONV)
0129
0130
                    CALL NEWTON(URDOT(1), VROOT(1), NQ, UQ

IF(CONV) GO TO 280

WRITE(102,1140) I, UROOT(1), VROOT(1)

IF(1.LT.,JAP) GO TO 241

IF(1.EQ.,JAP) GO TO 250

GO TO 300

241 KKK=JAP-1
0131
0132
0133
0134
0135
0136
0137
                          DO 245 II=I.KKK
                          UAP(11)=UAP(11+1)
0138
0139
                    245 VAP([])=VAP([]+1)
                    250 JAP=JAP-1
0140
                    250 JAP=JAP=1
GO TO 300
280 J1=J1+1
URGOTS(J1)=UZRO
VROOTS(J1)=VZRO
0141
0143
0144
0145
                    300 CONTINUE
                         IF(J1.EQ.0) GO TO 305
J=J1
0146
0147
0148
0149
                          00 303 [=1.J
                          UROUT(I)=UROUTS(I)
                    303 VROOT([] = VROOTS([]
0150
                    GD TO 307
305 WRITE(102,1150) NQ
0151
0152
0153
0154
                          KKK=NQ+1
                          NNN=KKK+1
                          DO 306 L=1.KKK
```

```
"'DT56
                                 JJJ=NNN-L
                         306 WRITE(102,1040) GNAME, ENTRY(JJJ), UQ(JJJ), VQ(JJJ)
0157
0158
                                GD TO 10
 0159
                          307 NEWT=.TRUE.
0160
                          310 CALL MULTIENP, UP, VP, J, URGOT, VROOT, MULTI
 0161
                                 IF(NEWT) GO TO 330
                                 WRITE1102-10701
 0162
                                WRITE(102,1165) (L,UROOT(L), VROOT(L), MULT(L), L=1, J)
 0163
                         GO TO 10
330 WRITE(102,1180)
 0164
 0165
                                WRITE(102,1190) (L,UROOT(L), VROOT(L), MULT(L), UAP(L), VAP(L), L=L, JAP
 0166
                                KKK=JAP+1
 0167
                                IF JAP. LT.J) WRITE(ID2, L165) (L, UROGT(L), VROGT(L), MULT(L), L=KKK, J)
 0168
 0169
 0170
                        1000 FORMAT(3112,1X1,9X,13,1X,4(06.0,1X),13X,2(D7.0,1X),I1)
 0171
                        1010 FORMAT(2030.0)
 0172
                        1015 FORMAT42030-0)
                        1020 FORMAT(1H1,10X,41HGREATEST COMMON DIVISOR METHOD USED WITH +2(A4).
 0173
                              135HMETHOD TO FIND ZEROS OF POLYNOMIALS/11x, 18HPOLYNOMIAL NUMBER . 1
                        1030 FORMAT(1X, 22HTHE DEGREE OF P(X) IS .12,22H THE COEFFICIENTS ARE//
 0174
                       1)
1040 FORMAT(2X,A2,A2,AH) = ,D23,16,3H + ,D23,16,2H 1)
1070 FORMAT(///1X,13HROOTS OF P(X),52X,14HMULTIPLICITIES//)
1080 FORMAT(2X,5HROOT(,12,4H) = ,D23,16,3H + ,D23,16,2H 1,10X,12)
1100 FORMAT(2X,A3,A2,4H) = ,D23,16,3H + ,D23,16,2H 1)
1120 FORMAT(//1X,73HQ(X) 15 THE POLYNOMIAL HHICH HAS AS ITS ROOTS THE 1D1STINCT ROOTS OF P(X),/1X,22HTHE DEGREE OF Q(X) IS ,12,22H THE C 20EFFICIENTS ARE//)
1200 FORMAT(//1X,74DQCGEETCIENTS OF THE DEGLATED POLYNOMIAL FOR WHICH
 0175
 0176
 0177
 0178
 0179
                        1200 FORMAT(///1x,70HCOEFFICIENTS OF THE DEFLATED POLYNOMIAL FOR WHICH ING ZERGS WERE FOUND.//)
0180
                        1132 FORMAT(///1x,13HROOTS OF Q(x),84x,21HIN(TIAL APPROXIMATION//)
 0181
                        1133 FORMAT(2x,5HROOT(,12,4H) = ,D23.16,3H + ,D23.16,2H 1,17x,D23.16,3H
 0182
                              1 + .D23.16.2H I)
                       1134 FORMAT(2X,5HROOT(,12,4H) = ,D23.16,3H + ,D23.16,2H I,20X,26HRESULT 1S OF SUBROUTINE QUAD)
 0183
 0184
                        1140 FORMATI///, 1X, 40 HNO ROOTS FOR INITIAL APPROXIMATION ROOTE, 12, 4H) = 1
                       140 FORMATI///,1X,40HNO ROOTS FOR INITIAL APPROXIMATION ROOTE,12,4H) = 1,D23.16,3H + ,D23.16.2H [)

150 FORMATI///,1X,45HNO ROOTS FOR THE POLYNOMIAL Q(X) OF DEGREE = ,12, 138H WITH GENERATED (NITIAL APPROXIMATIONS//)

1165 FORMATI2X,5HROOT (,12,4H) = ,D23.16,3H + ,D23.16,2H [,7X,12,10X,26H lresults of subroutine quad)

1180 FORMATI///1X,13HROOTS OF P(X),52X,14HMULTIPL(CITIES,17X,21HINITIAL)
 0185
 0186
0187
                              1 APPROXIMATION//I
 8810
                        1190 FORMATI2X,5HRDOT(,12,4H) = ,D23,16,3H + ,D23,16,2H 1,7X,12,7X,D23.
                       1190 FORMAT(12x,5) FROUT(12,4H) = ,023.16,3H + ,023.16,2H 1,7x,12.16,3H + ,023.16,2H 1)
2000 FORMAT(1x,4) HNUMBER OF INITIAL APPROXIMATIONS GIVEN. ,12)
2010 FORMAT(1x,29 HMAX[MUM NUMBER OF ITERATIONS.,11x,13)
2020 FORMAT(1x,21 HTEST FOR CONVERGENCE.,13x,09.2)
2030 FORMAT(1x,24 HTEST FOR MULTIPLICITIES.,10x,09.2)
2040 FORMAT(1x,23 HRADIUS TO START SEARCH.,11x,09.2)
2050 FORMAT(1x,21 HRADIUS TO END SEARCH.,13x,09.2)
2060 FORMAT(//IX)
2070 FORMAT(//IX)
 0189
 0190
 0191
 0192
 0193
 0194
 0195
                        2070 FORMATILX, 34HTEST FOR ZERO IN SUBROUTINE GCD. . D9.2)
0196
0197
                        2080 FORMAT(1X+34HTEST FOR ZERO IN SUBROUTINE QUAD- +D9-2)
0198
                                END
```

0001		ç	SUBROUTINE GENAPP(APPR,APPI,NAPP,XSTART)
	-	000	* SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE * DEGREE OF THE ORIGINAL POLYNOMIAL. *
4,		č	***************************************
0002	A		DOUBLE PRECISION APPR.APPI.XSTART.BETA. EPS1.EPS2.EPS3.EPS4
0003		•	DIMENSION APPR(25).APPI(25)
0004		- 5	COMMON EPS1, EPS2, EPS3, EPS4, IO2, MAX
0005	100		IF(XSTART.EQ.O.O) XSTART=0.5
0006			BETA=0.2617994
0007			00 10 [=1; NAPP
8000	17.5	•	APPRILI=XSTART*OCOS(BETA)
0009			APPI(1)=XSTART+DSIN(BETA)
0010			BETA=BETA+0.5235988
0011		. 1	O XSTART=XSTART+0.5
0012	. ,		RETURN
0013	23		END THE RESERVE OF THE PROPERTY OF THE PROPERT

```
SUBROUTINE ALTER(XOLDR, XOLDI, NALTER, ITIME)
0001
                                                                       000000
                                                                                                          SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT.
                                                                                                      DOUBLE PRECISION XQLDR, XQLDI, ABXQLD, BETA, EPS1, EPS2, EPS3, EPS4
COMMON EPS1, EPS2, EPS3, EPS4, IO2, MAX
[F(ITIME.NE.O) GO TO 5
1TIME =1
MRITE(IO2, 1010) MAX
[F(NALTER.EQ.O) GO TO 10
WRITE(IO2, 1000) XQLDR, XQLDI
GO TO 20
ABXQLQ =DSORT(IXQLDR = XQLDI DESS(I) ABXQLQ = IOSORT(IXQLDR = XQLDI DESS(I) ABXQLQ = IOSORT(I) ABXQLQ = IOSO
0002
 0003
 0004
 0005
 0006
 0007
 0008
 0009
 0010
                                                                                         10 ABXOLO=DSQRT((XOLDR*XOLDR)+(XOLD1*XOLD1))
 0011
                                                                                                            BETA=DATAN2(XOLD1,XOLDR)
                                                                                       BETA=DATANZIXULUI,XULUNI
WRITE(102,1020) XOLOR,XOLDI
20 NALTER=NALTER+1
IF(NALTER-GT-5) RETURN
GO TO (30,40,30,40,30].NALTER
30 XOLDR=-XOLOR
XOLDI=-XOLDI
 0012
0013
0014
 0016
 0017
                                                                                         GO TO 50

40 BETA=BETA+L.047L976

XOLDR=ABXOLD+DCOS(BETA)

XOLDI=ABXOLD+DSIN(BETA)
 0018
 0019
0020
0021
                                                                                          50 RETURN
                                                                             1000 FORMAT(1X,D23.16,3H + ,D23.16,2H I,10X,21HALTERED APPROXIMATION)
1010 FORMAT(///1X,54HNG CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
17ER ,13,12H ITERATIONS.//)
1020 FORMAT(/1X,023.16,3H + ,D23.16,2H I,10X,21HINITIAL APPROXIMATION)
 0023
 0024
0025
 0026
                                                                                                           END
```

```
0001
                       SUBROUTINE GCD(N,UR,VR,M,US,VS,MI,USS,VSS)
                C
                C
                     * GIVEN POLYNOMIALS PIX) AND DPIX) WHERE DEG. DPIX) IS LESS THAN DEG. * PIX), SUBROUTINE GCD COMPUTES THE GREATEST COMMON DIVISOR OF PIX) AND * DPIX).
                ç
                C
                        DOUBLE PRECISION USSSSSS VSSSSS DOUBLE PRECISION UR, VR, US, VS, USS, VSS, URR, VRR, UD, VD, UT, VT, EPSLON, EP
0002
0003
                      152, EP$3, EP$4, BBB
0004
                       DIMENSION UR(26), VR(26), US(26), VS(26), USS(26), VSS(26), URR(26), VRR(
                      126), UT(26), VT(26)
0005
                       COMMON EPSLON, EPS2, EPS3, EPS4, 102, MAX
8000
                       N1=N
0007
                       M1=M
0008
                       KKK=N+1
0009
                       DO 20 1=1.KKK
0010
                       (1) AU=(1) AAU
0011
                   20 VRR(I)=VR(I)
                       KKK=M+1
DD 25 I=1+KKK
USS(I)=US(I)
0013
0014
0015
                       V$$(11=V$(11
0016
                   30 BBB=USS(MI+1)*USS(MI+1)+VSS(M1+1)*VSS(M1+1)
0017
                       UD=(URR(N1+1)+USS(M1+1)+VRR(N1+1)+VSS(M1+1))/BBB
0018
                       VD=(USS(ML+1)*VRR(N1+1)-URR(N1+1)*VSS(M1+1))/BBB
0019
                       KKK=N1+1-M1
0020
                       DO 40 I=KKK, N1
                      UT(1)=URR(1)-(UD*USS(1-N1+M1)-VD*VSS(1-N1+M1))
VT(1)=VRR(1)-(UD*VSS(1-N1+M1)+VD*USS(1-N1+M1))
IF(M1.EQ.N1) GD TO 70
KKK=N1-M1
0021
0022
0023
0024
0025
                       DO 60 [=1.KKK
0026
                       UT(1)=URR([)
0027
                      VT(I)=VRR(I)
                   70 DO 90 [=1,N] 8BB=DSQRT(UT(N1+1-1)*UT(N1+1-1)*VT(N1+1-1)*
0028
0029
0030
                       (F(BBB.GT.EPSLON) GO TO 100
0031
                      CONTINUE
0032
                       B0B=USS(M1+1)+USS(M1+1)+VSS(M1+1)+VSS(M1+1))
USSSSS=(USS(()+USS(M1+1)+VSS(())+VSS(M1+1))/000
VSSSSS=(VSS(()+USS(M1+1)-USS(())+VSS(M1+1))/000
0033
0034
0035
0036
                       USS(1)=USSSSS
                      VSSIII=VSSSSS
0037
0038
                       USS(M1+1)=1.0
                       VSS(M1+1)=0.0
0039
                  GO TO 200
100 K=N1-1
0040
0041
0042
                       IF(K.EQ.O) GO TO 170
                       IF(K-LT-H1) GO TO 140
0043
0044
                       KKK=K+1
                      00 130 J=1,KKK
URR(J)=UT(J)
0045
0046
                 130 VRR(JI=VT(J)
0047
0048
                       N1=K
                       GO TO 30
0049
```

0050 1	40 KKK=K+1
0051	DO 150 J=1.KKK
0052	URR(J)=USS(J)
0053	VRR(J)=VSS(J)
0054	USSIJ)*UT(J)
0055	(L)TV=(L122V 06
0056	KKK=K+2
0057	NNN=M1+1
0058	DO 160 JEKKK-NNN
0059	URR(J)=USS(J)
0060	(L)22V=[L) RRV 00
0061	N1=M1
0062	M1=K
0063	GO TO 30
	70 USS111=1.0
0065	VS5{1)=0.0
0066	MI=0
0067 20	O RETURN
0068	END

```
SUBROUTINE QUAD(N,UA,VA,J,UROOT,VROOT,MULT)
0001
                     0000
                               SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR. SOLUTION OF THE QUADRATIC IS DONE USING THE QUADRATIC FORMULA.
                               DOUBLE PRECISION UA. VA. URGOT, VROOT, UDISC, VDISC, UTEMP, VTEMP, UD. VD.E
0002
                              1PS1, EPS2, EPS4, EPSLON, 88B
                               DIMENSION UA(26), VA(26), UROOT(25), VROOT(25), MULT(25)
COMMON EPSI, EPS2, EPSLON, EPS4, IO2, MAX
0003
0004
                                IF(N.GT.1) GO TO 60
IFIJ.LT.0) GO TO 40
0005
0006
0007
0008
                                J = J + 1
                                GO TO 50
0009
                          40 MULTILI=1
0010
0011
                           50 BBB=UA(2)+UA(2)+VA(2)+VA(2)
                                UROOT(J)=-(UA(1)*UA(2)*VA(1)*VA(2))/BBB
VROOT(J)=-(VA(1)*UA(2)-UA(1)*VA(2))/BBB
0012
0013
                          VROOT(J) == {VA[[] *UA(Z] -UA(I] *VA(Z]) / BBB
GO TO 200
60 UDISC={UA(Z] *UA(Z) -VA(Z) *VA(Z)] - (4.0*(UA(3) *UA(1) -VA(3) *VA(1)) }
VDISC={2.0*UA(Z] *VA(Z)] - (4.0*(UA(3) *VA(1) +VA(3) *UA(1)) }
BBB=DSQRTHUDISC*UDISC*VDISC*VDISC*)
IF(BBB.LE.EPSLON) GO TO 100
IF(J.GE.O) GO TO 80
MULT(1) = 1
0014
0015
0016
0017
0018
0019
0020
                                MULT(2)=1
0022
                               CALL COMSQTEUDISC. VDISC. UTEMP. VTEMP)
0023
                                UD=2.0+UA(3)
0024
                                VD=2.0+VA131
0025
                                888=UD*UD+VD*VD
0026
                                BBB=UJ*UD*VD*VD
UROOT[J+1]=({-UA(2)+UTEMP]*UD+(-VA(2)+VTEMP)*VD)/88B
VROOT[J+1]=({-VA(2)+VTEMP}*UD-(-UA(2)+UTEMP)*VD)/BBB
UROOT[J+2]=((-UA(2)-UTEMP)*UD+(-VA(2)-VTEMP)*VD)/88B
VROOT[J+2]=((-VA(2)-VTEMP)*UD-(-UA(2)-UTEMP)*VD)/88B
0027
0028
0029
0030
                                J=J+2
0031
0032
                                GO TO 200
0033
                         100 [f(J.LT.0) GD TO 110
0034
                                J=J+1
                         GO TO 130
0035
0036
                                J=1
0037
                         130 UD=2.0*UA(3)
0038
0039
                                VD=2.0*VA(3)
                                BBB=UD+UD+VD+VD
UROOT[J]=(~UA[2]+UD-VA(2]+VD)/BBB
0040
0041
                                VRDOT(J)=(-VA(2)+UD+UA(2)+VD)/BBB
0042
                                RETURN
0043
0044
```

```
SUBROUTINE NEWTONIUX, VX. N. UP. VP. UXG. VXG. CONVI
0001
                    00000000
                              THIS SUBROUTINE CALCULATES A NEW APPROXIMATION FROM THE OLD APPROXIMATION BY USING THE (TERATION FORMULA x(n+1) = x(n) - p(x(n)) / p^*(x(n)).
                             DOUBLE PRECISION UX, VX, UP, VP, UXQ, VXQ, UB, VB, UDPXQ, VDPXQ, UPXQ, VPXQ, U
10[FF, VD1FF, EPS1, EPSLON, EPS3, EPS4, AAA, BBB
DOUBLE PRECISION DDD
DOUBLE PRECISION ABPXQ
DIMENSION UP(261, VP(261, UB(261, VB(26))
COMMON EPS1, EPSLON, EPS3, EPS4, 102, MAX
LOGICAL CONV
0002
0003
0004
0005
0006
0007
0008
                               XU=0XU
0009
                               XV=OXV
                               DO LO 1=1,MAX
CALL HORNER(UXO, VXO,N,UP, VP,UB, VB,UOPXO, VDPXQ)
UPXO=UB(1)
0010
0011
0012
                               VPXO=VB(1)
0013
                               DOD=OSQRT(UDPXD*UDPXQ+VDPXO*VDPXO)
0014
                               IF(DDD.NE.0.0) GO TO 5
ABPXO=DSQRT(UPXO+UPXO+VPXO+VPXO)
IF(ABPXO.EQ.0.0) GO TO 20
0015
0016
0017
                               GO TO 15
0018
0019
                            5 BBB=UDPXO+UOPXO+VDPXG+VDPXO
                               UDIFF=(UPXO*UDPXO+YPXO*YDPXO)/8B8
VDIFF=(VPXO*UDPXO-UPXO*YDPXO)/8B8
0020
0021
                               UXO=UXO-UDIFF
VXO=VXO-VDIFF
AAA=DSORT(UDIFF*UDIFF+VDIFF*VOIFF)
0022
0023
                               BBB=DSQRT(UXQ+UXQ+VXQ+VXQ1
0025
                          IF(888.E0.0.0) GO TO 10
1F(AAA/888.LT.EPSLGN) GO TO 20
10 CONTINUE
0026
0027
0028
                          15 CONV=.FALSE.
0029
0030
                               RETURN
                               CONV=.TRUE.
0031
0032
                               RETURN
0033
                               END
```

```
. 0001
                                   SUBROUTINE DIVIDEIN. UP. VP.M. UD. VD. K. UQ. VQ!
                        000000
                                   GIVEN TWO POLYNOMIALS FIX) AND GIX). SUBROUTINE DIVIDE COMPUTES THE QUOTIENT POLYNOMIAL H(X) = F(X)/G(X).
                               **********************************
                                   DOUBLE PRECISION UP, VP, UD, VD, UQ, VQ, UTERM, VTERM, UDUMMY DIMENSION UP(26), VP(26), UD(26), VD(26), UQ(26), VQ(26)
   0002
   0003
   0004
                                   AGIK+1)=(Abi H+1)+nDi H+1)-nbiH+1)+ADiH+1)3\NDNHAA
NDIK+1)=(AbiH+1)+nDiH+1)+AbiH+1)+ADiH+1)3\NDNHAA
NDNHAA-NDIH+1)+NDIH+1)+ADIH+1)+ADIH+1)
   0005
  0006
0007
0008
                                   1F(K.EQ. 0) 60 TO 100
   0009
  0010
0011
0012
0013
                                   00 50 I=1.K
                                   J=J+1
UTERM=UP(N-J)
VTERM=VP(N-J)
  0014
0015
                                   KK=K+1
                                   NNN=M-J
DD 40 H1=NNN+M
IF(KK.GT.1) GD TO 10
   0016
  0017
0018
0019
0020
                                 GO TO 45

1F(ML.GE.1) GO TO 20

GO TO 40

UTERM-UTERM-(UQ(KK)*UD(M1)-VQ(KK)*VO(M1))
VTERM-VTERM-(UQ(KK)*VO(M1)+VQ(KK)*UD(M1))
   1500
   0022
   0023
                                   KK=KK-L
                             SO VQ(K+1-I)=(VTERM+UD(M+1)+VD(M+1)+VD(M+1))/UDUMMY
50 VQ(K+1-I)=(VTERM+UD(M+1)+VTERM+VD(M+1))/UDUMMY
  0024
  0026
  0027
                            100
                                   RETURN
   0028
                                   €ND
```

```
0001
                                         "SUBROUTINE HORNER (UX, VX, N, UP, VP, UB, VB, UC, VC)
                            0000000
                                      * HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(x) AT A * POINT D AND ITS DERIVATIVE AT D. SYNTHETIC DIVISION IS USED TO * DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE FACTOR \{x-0\}.
                                          DOUBLE PRECISION UX, VX, UP, VP, UB, VB, UC, VC
DOUBLE PRECISION UDUMMY, VDUMMY
DIMENSION UP(26), VP(26), UB(26), VB(26)
UB(N+1)=UP(N+1)
VB(N+1)=VP(N+1)
UB(N)=(UX*UB(N+1)-VX*VB(N+1))+UP(N)
VB(N)=(UX*UB(N+1)+VX*UB(N+1))+VP(N)
 0002
 0003
 0004
0005
 0007
 0008
 0009
                                           UC=UB{N+1}
0010
0011
0012
                                           VC=VB(N+1)
                                          KKK=N-1
00 10 [=1,KKK
                                          \text{\def{kkk+1-1} = (\text{\def{kkk+2-1}}-\text{\def{kkk+2-1}} \) + \text{\def{kkk+1-1}} 
\text{\def{kkk+1-1} = (\text{\def{kkk+2-1}}+\text{\def{kkk+2-1}} \) + \text{\def{kkk+1-1}}
0013
 0015
                                           UDUMMY=UX+UC-YX+VC
0016
                                           VDUMMY=UX+VC+VX+UC
                                         AC=ADMWA.+AB (KKK+5-1)
AC=ADMWA.+AB (KKK+5-1)
0017
0018
                                           RETURN
0020
                                           END
```

```
SUBROUTINE DERIVIN, UP, VP, M, UA, VA)
1000
                 00000
                      * GIVEN A POLYNOMIAL PIXE, SUBROUTINE DERLY COMPUTES THE COEFFICIENTS OF * ITS DERIVATIVE P^*(X).
                         DOUBLE PRECISION UP, VP, UA, VA, AAA
DIMENSION UP(26), VP(26), UA(26), VA(26)
0002
0003
0004
                         KKK=N+1
                         00 10 1=2.KKK
AAA=1-1
UA(1-1)=AAA+UP(1)
0005
0007
0008
                         VA(I-1)=AAA+VP(I)
0009
                         M=N-1
0010
                         RETURN
0011
                         FND
```

```
SUBROUTINE MULTICA, UP, VP, J, UROOT, VROOT, MULT)
0001
                C
                Ç
                         GIVEN N ZEROS OF A POLYNOMIAL, SUBROUTINE MULTI COMPUTES THEIR
                Č
                         MULTIPLICITIES.
                C
                         DOUBLE PRECISION UP, VP, URDOT, VROOT, UA, VA, UB, VB, UC, VC, EPS1, EPS2, EPS
0002
                       1LON, EP$3,888
                         DIMENSION UP(26), VP(26), UROOT(25), VROOT(25), UA(26), VA(26), UB(26), V
0003
                        18(26), MULT(25)
                         COMMON EPS1, EPS2, EPS3, EPSLON, TO2, MAX
DO 100 F=1, J
KKK=N+1
0004
0005
0006
                         00 10 K#1.KKK
                         UALKI=UPLKI
8000
0.009
                     10 VA(K)=VP(K)
0010
                         M=N
                    MULT(1)=0
20 CALL HORNER(UROOT(!), VROOT(!), M, UA, VA, UB, VB, UC, VC)
0011
0012
0013
                         BBB=DSQRT(UB(1) *UB(1) +VB(1) *VB(1))
                         [FIBBB.LT.EPSLON] GO TO 50
0014
0015
                         IF(MULT(I).EQ.D) GO TO 40
                     GO TO 100
40 WRITE(102,1000) EPSLON, (, URGOT(1), VRODT(1)
0016
0017
                     GO TO 100
50 MULT(1)+MULT(1)+1
0018
0019
                    [FIM.GT.1] GO TO 60
GO TO 100
60 DO 70 K=1.M
0020
0021
0022
                        UA(K)=UB(K+1)
VA(K)=VB(K+1)
0023
0024
                         M=M-I
0025
                   GO TO 20
100 CONTINUE
0026
0027
                  RETURN

1000 FORMAT(///15H THE EPS(LON 1.D10.3.48H) CHECK IN SUBROUTINE MULTI

11NDICATES THAT ROOT(.12.4H) = .D23.16.3H + .D23.16.2H 1./80H IS NO
27 CLOSE ENOUGH TO BE A TRUE ROOT. IT IS PRINTED BELOW WITH MULTIP
0028
0029
                        BLICITY 0//)
0030
                         END
```

```
0001
                                -SUBROUTINE-CONSQUUX,YX,UY,YY)
                      00000
                              * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.
                           DOUBLE PRECISION UX, VX, UY, VY, DUMMY, R. AAA. BBB
R=DSQRT(UX+UX+VX+VX)
AAA=DSQRT(DABS((R+UX)/2.0))
BBB=DSQRT(DABS((R-UX)/2.0))
IF(VX) 10,20,30
10 UY=AAA
VY=-1.0+8BB
GO TO 100
20 IF(UX) 40,50,60
30 UY=AAA
VY=BBB
GO TO 100
                              ************
 0002
 0003
 0004
0005
 0006
 0007
 0008
 0009
0010
0011
 0012
                           GO TO 100
40 DUMMY=DABS(UX)
 0013
 0014
 0015
                                 UY=0.0
                           YY=DSQRT(DUMMY)
GO TO 100
50 UY=0.0
VY=0.0
0016
0017
 0018
 0019
                         VY=0.0
GO TO 100
60 DUMMY=DABS(UX)
UY=DSQRT(DUMMY)
VY=0.0
LOO RETURN
END
 0020
0021
0022
0023
 0024
 0025
```

#### APPENDIX F

### G.C.D. - MULLER'S METHOD

### 1. Use of the Program

A double precision FORTRAN IV program using the G.C.D. method with Muller's method as a supporting method is presented here. Flow charts for this program are given in Figure F.1 while Table F.III gives a FORTRAN IV listing of this program. Single precision variables are listed in Table F.II. The single precision variables are used in the flow charts and the corresponding double precision variables can be obtained from Table F.II.

This program is designed to solve polynomials having degree less than or equal to 25. In order to solve polynomials of degree N where N > 25, the data statement and array dimensions given in Table F.I must be changed.

In this program both the leading coefficient and the constant coefficient are assumed to be non-zero.

### TABLE F.I

# PROGRAM CHANGES NECESSARY TO SOLVE POLYNOMIALS OF DEGREE GREATER THAN 25 BY G.C.D. - MULLER'S METHOD

#### Main Program

Subroutines MULTI, DIVIDE, DERIV, GCD, and QUAD

See corresponding subroutines in Table E.I.

### Subroutine MULLER

UROOT(N), VROOT(N)
MULT(N)
UAPP(N,3), VAPP(n,3)
UWORK(N+1), VWORK(N+1)
UB(N+1), VB(N+1)
UA(N+1), VA(N+1)
URAPP(N,3), VRAPP(N,3)

### Subroutine BETTER

UROOT(N), VROOT(N)
UA(N+1), VA(N+1)
UBAPP(N,3), VBAPP(N,3)
UB(N+1), VB(N+1)
UROOTS(N), VROOTS(N)
URAPP(N,3), VRAPP(N,3)
MULT(N)

Subroutine GENAPP

APPR(N,3) APPI(N,3)

#### Subroutine HORNER

UA(N+1), VA(N+1) UB(N+1), VB(N+1)

### 2. Input Data for G.C.D. - Muller's Method

The input data for G.C.D. - Muller's method is prepared exactly as described in Appendix E, § 2 for G.C.D. - Newton's method.

### 3. Variables Used in G.C.D. - Muller's Method

The main variables used in G.C.D. - Muller's method are given in Table F.II. The symbols used to indicate type and disposition are described in Appendix E, § 3. For variables not listed in Table F.II, see the main program or corresponding subprogram of Table E.VI.

### 4. Description of Program Output

The output from G.C.D. - Muller's method is identical to that for G.C.D. - Newton's method as described in Apptendix E, § 4, keeping in mind that Muller's instead of Newton's method is used. The expression "SOLVED BY DIRECT METHOD" is equivalent to "RESULTS OF SUBROUTINE QUAD." Only one initial approximation,  $X_0$ , (not three) is printed. The other two required by Muller's method were .9 $X_0$  and 1.1 $X_0$ .

### 5. Informative Messages and Error Messages

The informative messages and error messages in this program are described as follows. For other messages not listed here, see Appendix E, § 5.

"THE EPSILON (XXX) CHECK IN SUBROUTINE MULTI INDICATES THAT ROOT

YY = ZZZ IS NOT CLOSE ENOUGHT TO BE A TRUE ROOT. IT IS PRINTED BELOW

WITH MULTIPLICITY 0." This message is described in Appendix E, § 5.

"COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND." This message is described in Apptendix E, § 5.

"NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER XX." XX represents the number of the polynomial for which no zeros were extracted.

"IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT XX = YYY DID NOT CONVERGE AFTER ZZZ ITERATIONS." This message indicates that a root did not produce convergence during the attempt to improve accuracy. XX represents the number of the root before the attempt to improve accuracy, YYY represents its value, and ZZZ represents the maximum number of iterations. The following message then follows. "THE PRESENT APPROXIMATION IS AAA." AAA represents the present approximation to the root after the maximum number of iterations.

TABLE F.II

VARIABLES USED IN G.C.D. - MULLER'S METHOD

Single Precision Variable Type		Double Precision Variable Type		Disposition of Argument		
				Subro	outine MULLER	
NP	I	NP	I	E	Degree of polynomial P(X)	
NROOT	I	NROOT	I	R	Number of distinst roots found	
NOMULT	I.	NOMULT	Ι	•	Number of roots (counting multiplicities)	
ROOT	С	UROOT, VROOT	D	R	Array containing the roots	
NAPP	I	NAPP	I	E	Number of initial approximations to be read in	
APP	C	UAPP, VAPP	D	E	Array of initial approximations	
WORK ·	C	UWORK, VWORK	· • D		Working array containing coefficients of current polynomial	
В	C	UB, VB	D		Array containing coefficients of deflated polynomial	
A	C	UA, VA	D	${f E}$ .	Array containing coefficients of original polynomial, P(X)	
RAPP	С	URAPP, VRAPP	D	R	Array of initial or altered approximation for which	
					convergence was obtained	
X1	C	UX1,VX1	D		One of three current approximations to a root	
X2	С	UX2,VX2	D		One of three current approximations to a root	
Х3	C	UX3,VX3	D		One of three current approximations to a root	
PX1	С	UPX1, VPX1	D		Value of polynomial P(X) at X1	
PX2	C	UPX2, VPX2	D		Value of polynomial P(X) at X2	
PX3	C	UPX3,VPX3	D		Value of polynomial P(X) at X3	
X4	С	UX4,VX4	. D		Newest approximation $(X_{n+1})$ to root	
PX4	C	UPX4,VPX4	D		Value of polynomial P(X) at X4	
MULT	I	MULT	I	•	Array containing the multiplicities of each root found	
ITER	I	ITER	I		Counter for iterations	
101	I	I01	I		Unit number of input device	
102	I	102	I	<b>C</b> ·	Unit number of output device	
EPSRT	R	EPSRT	D	С	Number used in subroutine BETTER to generate two approximations from the one given	
NOPOLY	I	NOPOLY	I	E	Number of the polynomial	

TABLE F.II (Continued)

			recision Type	Disposition of Argument	<u>Description</u>
MAX	I	MAX	. 1	С	Maximum number of iterations
EPS	R		D	C	Tolerance check for convergence
EPSO	R	EPSO	D	C	Tolerance check for zero (0)
EPSM	·R	EPSM	D	C	Tolerance check for multiplicities
KCHECK	ľ	KCHECK	I		Program control, KCHECK = 1 stops execution of program
XSTART	R		D	<b>E</b> ·	Magnitude at which to start generating initial approximations
XEND	R	XEND	D	E	Magnitude at which to end generating initial approximations
NWO RK	I	NWORK	I		Degree of current deflated polynomial whose coefficients are in WORK
ITIME	I	ITIME	I		Program control
NALTER	I	NALTER	I		Number of alterations which have been performed on an initial approximation
IAPP	I	IAPP	I.		Counter for number of initial approximations used
CONV	L	CONV	L·		When CONV is true, convergence has been obtained
IROOT	I	IROOT	I	R	Number of distinct roots solved by Muller's method,
					i.e. not solved directly by subroutine QUAD
				Subre	outine HORNER
A	C	UA, VA	D	E	Array of current polynomial coefficients (to be deflated or evaluated)
NA	. I	NA	I	$\mathbf{E}_{-\epsilon}$	Degree of polynomial to be deflated or evaluated
X	С	UX,VX	D	E	Approximation at which to evaluate or deflate the polynomial
В	С	UB, VB	D	R	Array containing the coefficients of the deflated polynomial
PX	С	UPX, VPX	D	$\mathbf{R}$	Value of the polynomial at X
NUM	I	NUM	I		Number of coefficients of polynomial to be deflated

TABLE F.II (Continued)

Single Precision		Double Precision Disposition		sposition	
Variable	Туре	Variable <u>T</u>	ype of	Argument	Description
	<del></del>		,		
				Subr	coutine TEST
х3	С	UX3,VX3	D	E	Approximation to root (old) (X <sub>n</sub> )
X4	С	UX4,VX4	D	E	New approximation to root $(X_{n+1})$
CONV	L	CONV	L	R	CONV = True implies convergence has been obtained
EPS	R	EPS	D	С	Tolerance for convergence test
EPSO	R	EPSO	D	С	Tolerance check for zero (0)
DENOM	R	DENOM	D		Magnitude of new approximation, $(X_{n+1})$
				Subrout	ine BETTER
				•	•
MULT	I	MULT	I,	ECR	Array of multiplicities of each root
A	С	UA, VA	D	E	Array of coefficients of original undeflated polynomial
NP	I	NP	1.	E	Degree of original polynomial
ROOT	C	UROOT, VROOT	D .	ECR	Array of ROOTS
NROOT	I	NROOT	I	ECR	Number of roots stored in ROOT
BAPP	C	UBAPP, VBAPP	D	E	Array of initial approximations (old roots)
IROOT	I	IROOT	I	ECR	Number of roots solved by the iterative process (Not QUAD)
ROOTS	С	UROOTS, VROOTS	D		Temporary storage for new (better) roots
$\mathbf{L}$	I	L	I		Number of roots found by BETTER
EPSRT	R	EPSRT	D	C	A small number used to generate two of the three
					approximations when given one
ITER	I	ITER	I	C	Counter for number of iterations
В	С	UB, VB	D		Array containing coefficients of deflated polynomial
X1	C ·	UX1,VX1	D		One of three approximations to the root
X2	C	UX2,VX2	D		One of three approximations to the root
х3	C	UX3,VX3	D		One of three approximations to the root
PX1	С	UPX1,VPX1	D ·		Value of polynomial (P(X)) at Xl
PX2	. C	UPX2, VPX2	D		Value of polynomial (P(X)) at X2
PX3	C	UPX3,VPX3	D		Value of polynomial (P(X)) at X3

TABLE F.II (Continued)

Single Pre Variable	cision Type	Double Pre Variable		Disposition of Argument	Description			
CONV	L·.	CONV	L		CONV = true implies convergence has been obtained			
X4	C	UX4,VX4	D		Newest approximation to root			
J.	I	J	Ι		Program control - counts the number of roots used as initial approximations			
MAX	I	MAX	I	С	Maximum number of iterations permitted			
102	I	102	I	C	Unit number of output device			
Subroutine ALTER								
X1	С	X1R,X1I	D	ECR	One of the three approximations to be altered			
X2	С	X2R,X2I	D	ECR	One of the three approximations to be altered			
х3	С	X3R,X3I	Ð	ECR	One of the three approximations to be altered			
X2R	R	X2R	D	•	Real part of complex approximation			
X2I	R	X2I	D		Imaginary part of complex approximation			

### Subroutine CALC

These variables are dummy variables used for temporary storage and thus, are not defined.

### MAIN PROGRAM

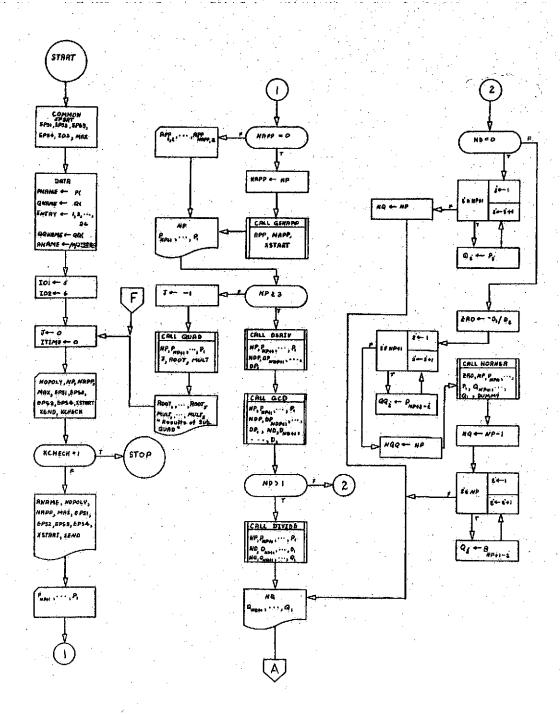


Figure F.1. Flow Charts for G.C.D.-Muller's Method

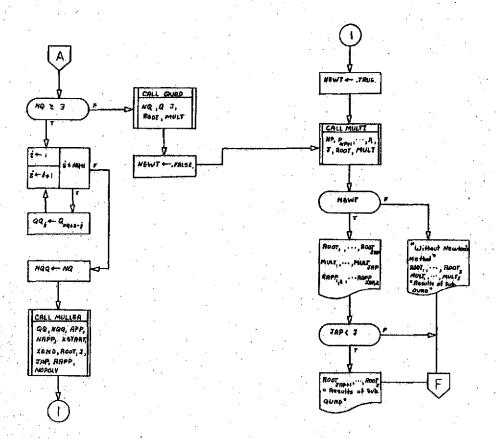


Figure F.1. (Continued)

### MULLER

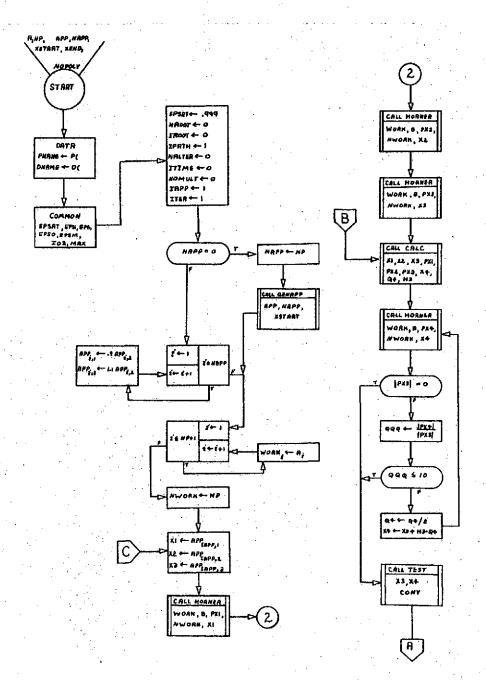


Figure F.1. (Continued)

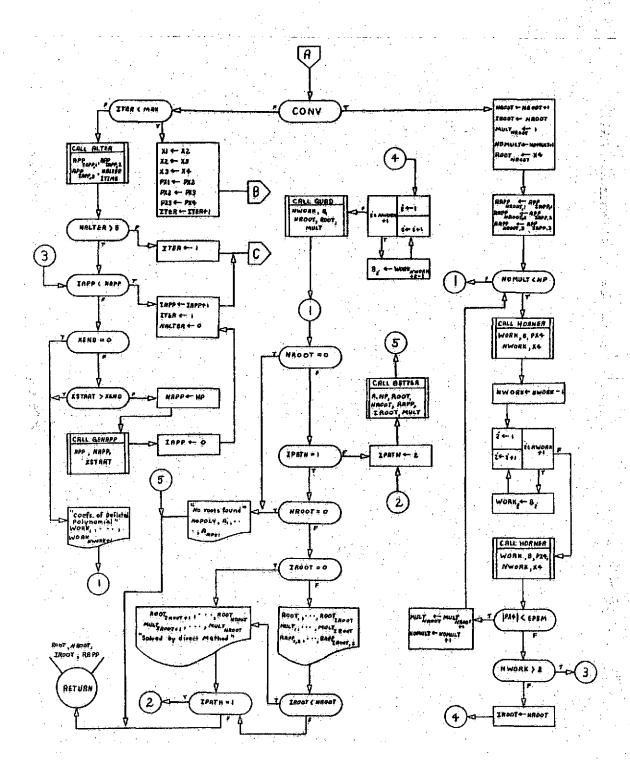


Figure F.1. (Continued)

## MULTI

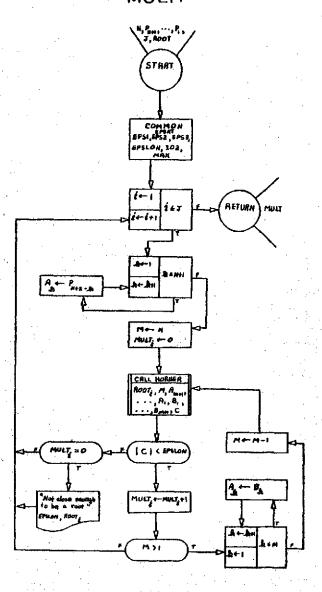


Figure F.1. (Continued)

# DIVIDE

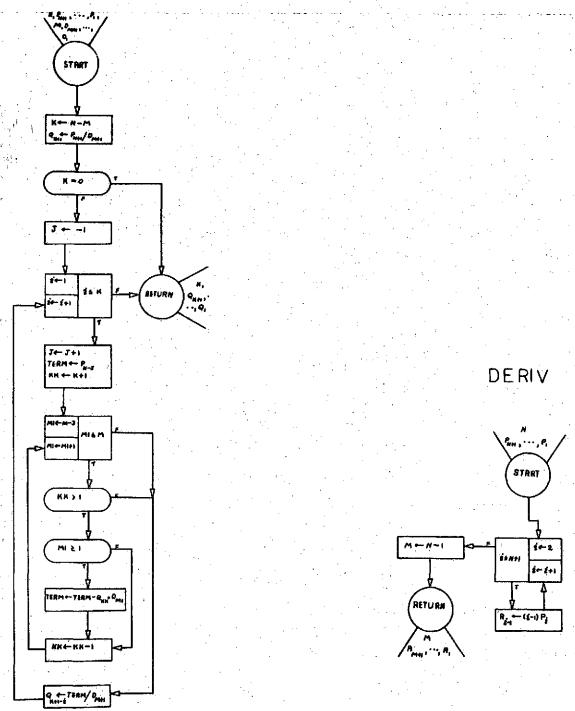


Figure F.1. (Continued)

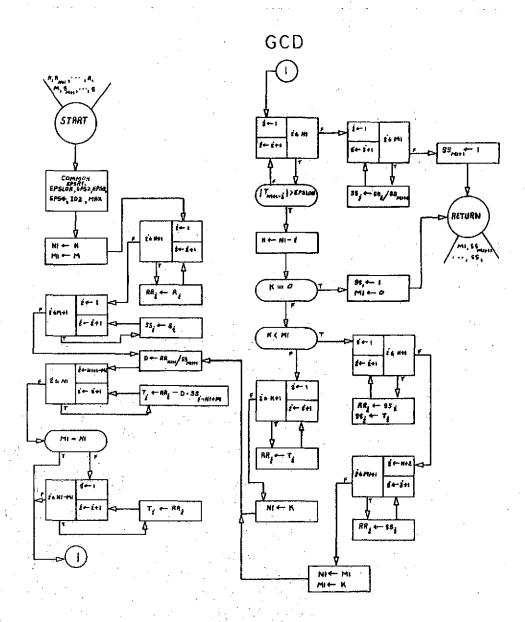


Figure F.1. (Continued)

## QUAD

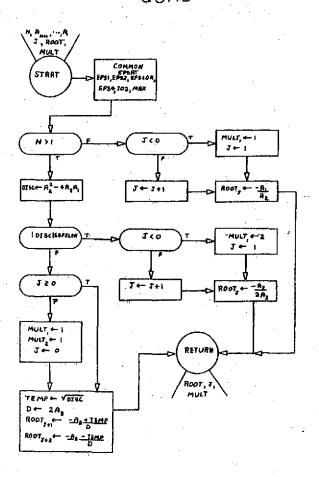


Figure F.1. (Continued)

BETTER

HORNER

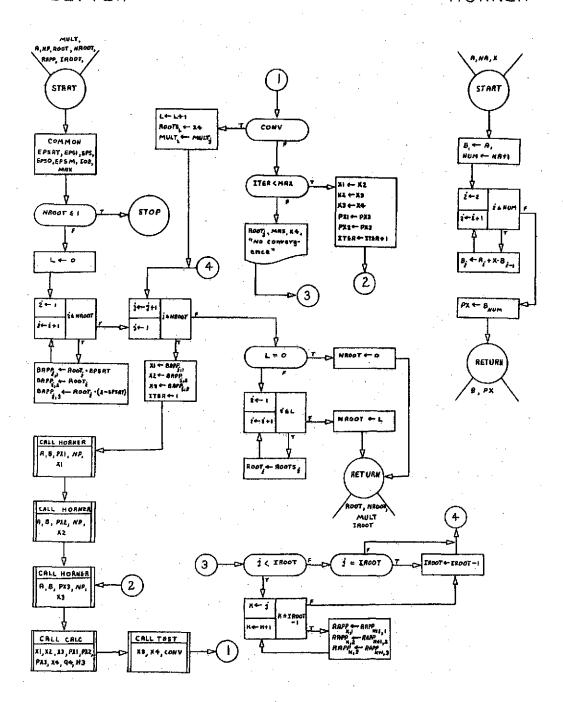


Figure F.1. (Continued)

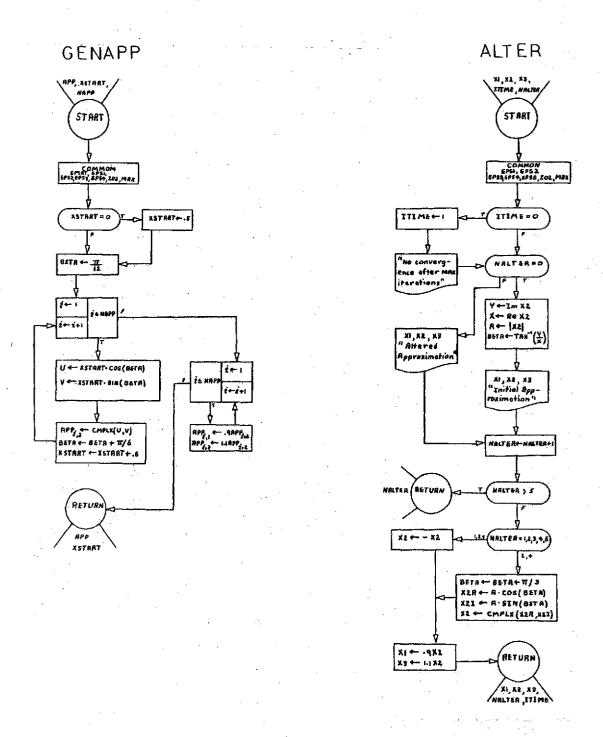


Figure F.1. (Continued)

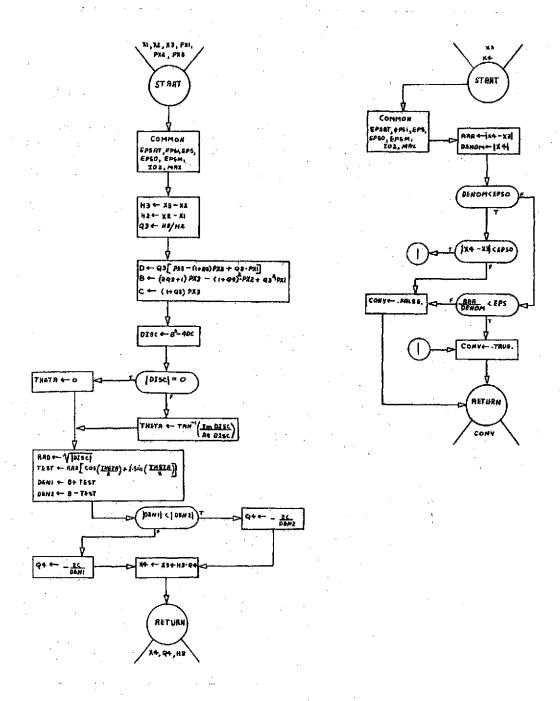


Figure F.1. (Continued)

## COMSQT

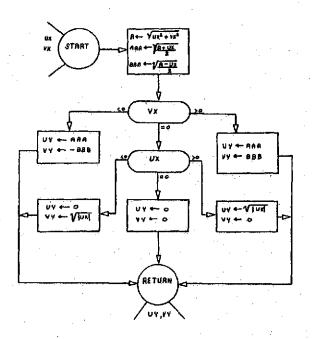


Figure F.1. (Continued)

### TABLE F.III

### PROGRAM FOR G.C.D.-MULLER'S METHOD

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

```
Č
                                               DOUBLE PRECISION PROGRAM FOR G.C.D. - MULLER'S METHOD
                               ם בי בי
                                               THE G.C.D. METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A
                                          * POLYNOMIAL OF MAXIMUM DEGREE 25. ALL MULTIPLE ROOTS ARE REMOVED BY

* DIVIDING THE POLYNOMIAL BY THE GREATEST COMMON DIVISOR OF THE POLYNOMIAL 4

* AND ITS DERIVATIVE. THE ZEROS OF THE RESULTING POLYNOMIAL ARE EXTRACTED 4

* AND THEIR MULTIPLICITIES DETERMINED.
                                ¢
                                                                   ********************
0001
                                                DOUBLE PRECISION URAPP, VRAPP
                                              DOUBLE PRECISION UP, VP, UAPP, VAPP, UROOT, VRODT, UDP, VDP, UD, VD, UZRO, VZ IRO, VQ, VD, UDUMMY, VOUMMY, VQQ, VQQ, UB, VB, EPS1, EPS2, EPS3, EPS4 DIMENSION URAPP(25, 31, VRAPP(25, 31, VAPP(25, 31, V
0002
0003
                                                DIMENSION UP(26), VP(26), URGOT(25), VROOT(25), MULT(25), UDP(26), VDP(2
0004
                                              16), UD(26), VD(26), UQ(26), VQ(26), UQQ(26), VQQ(26), UB(26), VB(26), ANAME
                                             212), ENTRY126)

DOUBLE PRECISION XSTART

DOUBLE PRECISION XEND

DOUBLE PRECISION EPSRT
0005
0006
0007
                                                COMMON EPSRT, EPS1, EPS2, EPS3, EPS4, TO2, MAX
ODOR
                                                DATA PNAME, QNAME, QQNAME/2HP1, 2HQ1, 3HQQ1/
0009
                                                DATA ENTRY/1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,2H10,2H11,2H12,2H13
0010
                                              1, 2H14, 2H15, 2H16, 2H17, 2H18, 2H19, 2H20, 2H21, 2H22, 2H23, 2H24, 2H25, 2H26/
                                                DATA ANAME(1), ANAME(2)/4HMULL, 4HERS /
0011
                                                LOGICAL NEWT
0012
                                                101=5
0014
                                                102=6
0015
                                        10 J=0
0016
                                                ITIME=0
                                                READ(ID1,1000) NOPOLY:NP:NAPP:MAX:EPS1:EPS2:EPS3:EPS4:XSTART:XEND;
0017
                                              IKCHECK
0018
                                                TELKCHECK.EQ.11 STOP
                                                WRITE(102,10201 ANAME(1), ANAME(21, NOPOLY
0019
0020
                                                WRITE(102,2000) NAPP
0021
                                                WRITE(102+2010) MAX
                                                WRITE(102,2070) EPS1
WRITE(102,2020) EPS2
WRITE(102,2080) EPS3
0022
0023
0024
0025
                                                WRITE(102,2030) EPS4
                                                WRITE(102,2040) XSTART WRITE(102,2050) XEND
0026
0027
                                                WRITE(102,2060)
0028
                                                KKK=NP+L
0029
                                                NNN=KKK+1
0030
                                                DO 20 1=1.KKK
0031
0032
                                                 1~444=ししし
                                       20 READ(101,1010) UP(JJJ), VP(JJJ)
1F(NAPP.NE.O) GO TO 22
0033
0034
                                                NAPP=NP
0035
                                                CALL GENAPP (UAPP, VAPP, NAPP, XSTART)
0036
0037
                                                GO TO 23
                                        22 READ([0], 1015] (UAPP([,2], VAPP([,2], I=1, NAPP)
0038
                                       23 WRITE(102,1030) NP
KKK=NP+1
0039
0040
                                                NNN=KKK+1
0041
```

```
00 25 1=1.KKK
0042
                     1-NNN=LLL
0043
0044
                  25 WRITE(102,1040) PNAME, ENTRY(JJJ1, UP(JJJ), VP(JJJ)
0045
                      IF(NP.GE.3) GO TO 30
0046
                     CALL QUADINE, UP. VP. J. URGOT, VROOT, MULT)
0047
0048
                     WRITE([02,1070]
                     WRITE(102,1165) (1,UROOT(1), VROOT(1), MULT(1), I=1,J)
0049
0050
                     GO TO 10
0051
                     CALL DERIVING, UP, VP, NOP, UDP, VDP)
                     CALL GCD(NP,UP,VP,NDP,UDP,VDP,ND,UD,VD)
IF(ND.GT.1) GO TO 70
IF(NO.EQ.O) GO TO 65
UDUMMY=UD(2)*UD(2)*VD(2)*VD(2)
0052
0053
0054
0055
0056
                     UZRD=-(UD(1)*UD(2)+VD(1)*VD(2))/UDUMMY
0057
                     V ZRO=-(UD(2)*VD(1)-UD(1)*VD(2))/UDUMMY
                     KKK=NP+1
0058
                     00 55 I=1,KKK
UQQ(I)=UP(KKK+1-I)
0059
0060
0061
                     VQQ(I)=VP(KKK+1-1)
0062
0063
                     CALL HORNER(NOQ, UQQ, VQQ, UZRO, VZRO, UB, V8, UDUMMY, VDUMMY)
                     NQ=NP-1
0064
                     00 60 f=1.NP
0065
                     UO(1)=UB(NP+1-1)
0066
0067
                     VQ(1)=VB(NP+1-1)
                     GO TO BO
0068
0069
                     KKK=NP+1
0070
                     DO 66 [=1,KKK
                     UQ(1)=UP(1)
0071
                     VOILLEVELLI
0072
                     NO=NP
0073
0074
                     G0 TO 80
                  TO CALL DIVIDEINP, UP, VP, ND, UD, VD, NQ, UQ, VQ)
0075
                 80 WRITE( [02, 1120) NO
0076
                     KKK=NQ+1
0077
0078
                     NNN=KKK+1
0079
                     00 83 I=1.KKK
                      ]ーがいれゃいし
0080
                 83 WRITE(102,1040) QNAME, ENTRY(JJJ), UQ(JJJ), VQ(JJJ)
1F(NQ.GE.3) GO.TO 85
1800
0082
                     GO TO 110
0083
                     KKK=NQ+1
0084
                     DO 90 I=1.KKK
0085
0086
                     UQQ(I)=UQ(KKK+1-I)
0087
                     VQQ(II=VQ{KKK+1~I}
0088
                     N00=N0
                     GO TO 120
0089
                LLO CALL QUADING.UQ, VQ. J. UROOT, VROOT, MULTI
0090
                     NEWT=.FALSE.
0091
0092
                     GO TO 310
                120 CALL MULLER(UQQ, VQQ, NQQ, UAPP, VAPP, NAPP, XSTART, XEND, UROOT, VROOT, J, J
1AP, URAPP, VRAPP, NOPOLY)
0093
                     NEWY = . TRUE .
0094
                310 CALL MULTI(NP.UP.VP.J.URGOT, VROOT, MULT)
0095
                     IF(NEWT) GO TO 330
0096
0097
                     WRITE( (02,1070)
0098
                     WRITE(102,1165) (L.UROGT(L), VROGT(L), MULT(L), L=1,J)
```

```
0099
                      GO TO 10
330 WRITE(102,1180)
0100
0101
                            DO 350 L=1.JAP
0102
                      350 WRITE(102,1190) L. URGOT(L), VRODT(L), MULTILI, URAPP(L,21, VRAPP(L,2)
0103
0104
                             IF(JAP.LT.J) WRITE([02,[165] [L.UROOT(L], VROOT(L), MULT(L), L=KKK, J)
                     GD TO 10
1000 FORMAT(3(12,1X),9X,13,1X,4(D6.0,1X),13X,2(D7.0,1X),11)
0105
0106
0107
                     1010 FORMAT (2030.0)
                     1015 FORMAT(2030.0)
0108
0109
                     1020 FORMATITHE, LOX, 41HGREATEST COMMON DIVISOR METHOD USED WITH , 2(A4),
                           135HMETHOD TO FIND ZEROS OF POLYNOMIALS/11X, 18HPOLYNOMIAL NUMBER +1
                    1030 FORMATIIX, 22HTHE DEGREE OF P(X) IS .12,22H THE COEFFICIENTS ARE//
0110
                    1040 FORMAT(2X,A2,A2,A4) = ,D23.16,3H + ,D23.16,2H 11

1070 FORMAT(///1X,13HROOTS OF PIX),52x,14HMULTIPLICITIES//)

1080 FORMAT(2X,5HROOT(,12,4H) = ,D23.16,3H + ,D23.16,2H (,10X,12)

1100 FORMAT(2X,A3,A2,4H) = ,D23.16,3H + ,D23.16,2H 1)

1120 FORMAT(///1X,73HQ(X) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE
0111
0112
0113
0114
0115
                           IDISTINCT ROOTS OF PIXI./IX,22HTHE DEGREE OF QIXI IS ,12,22H THE C
                           20EFFICIENTS ARE//)
0116
                    1165 FORMAT(2X,5HROOT(,12,4H) = ,023.16,3H + ,023.16,2H I_{*}7x,12,10x,26H
                    TRESULTS OF SUBROUTINE QUADI
118D FORMATI///IX.13HROOTS OF PIX).52X.14HNULTIPLICITIES.17X.21HINITIAL
0117
                           1 APPROXIMATION//1
0118
                     1190 FDRMAT(2x,5HRODT(,12,4H) = ,023,16,3H + ,023,16,2H 1,7x,12,9x,023,
                    1190 FDRMAT(2X,5H0DIT(12,4H) = ,023.16,3H + ,023.16,2H [,7X,12.116,3H + ,023.16,2H I)
2000 FDRMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN. ,12)
2010 FDRMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,13)
2020 FDRMAT(1X,21HTEST FOR CONVERGENCE.,13X,D9.2)
2030 FDRMAT(1X,24HTEST FOR MULTIPLICITIES.,10X,D9.2)
2040 FDRMAT(1X,23HRADIUS TO START SEARCH.,11X,D9.2)
0119
0120
0121
0122
0123
0124
                    2050 FORMAT (1X, 21HRADIUS TO END SEARCH., 13X, 09.2)
                    2060 FORMAT(//1X)
0125
                    2070 FORMAT(1X,34HTEST FOR ZERO IN SUBROUTINE GCD. .D9.2)
2080 FORMAT(1X,34HTEST FOR ZERO IN SUBROUTINE QUAD. .D9.2)
0126
0127
0128
```

```
0001
                         SUBROUTINE MULTITALUP, VP. J. URDOT, VROOT, MULT)
                        GIVEN N ZEROS OF A POLYNOMIAL, SUBROUTINE MULTI COMPUTES THEIR MULTIPLICITIES.
                C
                Ċ
0002
                        DOUBLE PRECISION UP. VP. UROOT, VROOT, UA. VA. UB. VB. UC. VC. EPS1. EPS2. EPS
                        ILON.EPS3.888
                        DIMENSION UP(26), VP(26), UROQT(25), VROOT(25), UA(26), VA(26), UB(26), VLB(26), MULT(25)
0003
0004
                        DOUBLE PRECISION EPSRT
0005
                         COMMON EPSRT.EPS1.EPS2.EPS3.EPSLON.102.MAX
0006
                        00 100 I=1,J
0007
                         KKK=N+1
                        DO 10 K=1,KKK
UA(K)=UP(KKX+1-K)
0008
0009
0010
                    10 VA(K)=VP(KKK+1-K)
0011
0012
                         HULT(1)=0
                    20 CALL HORNER(M,UA,VA,UROOT(I),VROOT(I),UB,VB,UC,VC)
BBB=OSORT(UC+UC+VC+VC)
0013
0014
                         IF(BB.LT.EPSLON) GO TO 50
0015
0016
                         IF(MULT(11.EQ.0) GO TO 40
0017
                         GO TO 100
                    40 WRITE(102,1000) EPSLON, I, URODT(1), VROOT(1)
0018
                    GO TO 100
50 MULT(I)=MULT(I)+1
IF(M.GT.1) GO TO 60
0019
0020
0021
                    GO TO 100
60 DO TO K=1,M
UA(K)=UB(K)
70 VA(K)=VB(K)
M=M~1
0022
0023
0024
0025
9200
0027
                        GO TO 20
0028
                   100 CONTINUE
0029
                        RETURN
                  1000 FORMAT(///15H THE EPSILON (,DIO.3.48H) CHECK IN SUBROUTINE MULTI
LINDICATES THAT ROOT(,12,4H) = ,D23,16,3H + ,D23,16,2H I,/80H IS NO
2T CLOSE ENDUGH TO BE A TRUE ROOT. IT IS PRINTED BELOW WITH MULTIP
0030
                        3LICITY 0//)
0031
```

```
1000
                         SUBROUTINE DIVIDE(N, UP, VP, N, UD, VD, K, UQ, VQ)
                C
                        GIVEN THO POLYNOMIALS F(X) AND G(X), SUBROUTINE DIVIDE COMPUTES THE QUOTIENT POLYNOMIAL H(X) = F(X)/G(X).
                Č
                C
                        DOUBLE PRECISION UP, VP, UD, VD, UQ, VQ, UTERM, VTERM, UDUMNY DIMENSION UP(26), VP(26), UD(26), VD(26), UQ(26), VQ(26)
0002
0003
0004
0005
                         UDUMMY=UD(M+1)*UD(M+1)+VD(M+1)*VD(M+1)
                         UQ(K+1)*(UP(N+1)*UD(M+1)+VP(N+1)*VO(M+1))/UDUMMY
VQ(K+1)*(UP(N+1)*UD(M+1)-UP(N+1)*VO(M+1))/UDUMMY
0006
0007
8000
                         IF(K.EQ.0) GO TO 100
0009
                         J=-1
0010
                         00 50 I=1.K
0011
                         J=J+1
0012
                         UTERM=UP(N-J)
                         VTERM= VP (N-J)
0013
0014
                         KK≂K+1
                         NNN=M-J
0015
0016
                         DO 40 MI=NNN.M
0017
                         1F(KK.GT.1) GO TO 10
0018
                         GO TO 45
                    10 IF(M1.GE.1) GO TO 20
GO TO 40
0019
0020
002 L
                     20 UTERH=UTERH-{UQ(KK}+UD(H] |-VQ(KK)+VD(H]))
0022
                         VTERM=VTERM-(UQ(KK)+VD(M1)+VQ(KK)+UD(N1))
0023
                     40 KK#KK-1
                    45 UDUMMY=UD(M+1)+UD(M+1)+VD(M+1)*VD(M+1)
UQ(K+1-1)=(UTERM*UD(M+1)+VTERM*VD(M+1))/UDUMMY
50 VQ(K+1-1)=(VTERM*UD(M+1)-UTERM*VD(M+1))/UDUMMY
0024
0025
0026
0027
                   100 RETURN
0028
0001
                         SUBROUTINE DERIVIN, UP, VP, M, UA, VA)
                C
                c
c
c
                        GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF
                        ITS DERIVATIVE P'(X).
                C
                        DOUBLE PRECISION UP, VP, UA, VA, AAA
DIMENSION UP1261, VP1261, UA(26), VA(26)
0002
0003
0004
                        KKK=N+1
                        DO 10 1=2,KKK
AAA*I-1
0005
0006
0007
                        UA(I-L)=AAA+UP(I)
                    10 VA(I-1)=AAA*VP(I)
M=N-1
8000
0009
0010
                         RETURN
0011
                        END
```

```
SUBROUTINE GCD(N,UR,VR,M,US,VS,MI,USS,VSS)
0001
             C
             C
                    GIVEN POLYNOMIALS PIX) AND DPIX) WHERE DEG. DPIX) IS LESS THAN DEG.
              Ē
                  * P(x), SUBROUTINE GCD COMPUTES THE GREATEST COMMON DIVISOR OF P(x) AND * DP(x).
              Č
              c
                       DOUBLE PRECISION EPSRT
 0002
 0003
                    DOUBLE PRECISION USSSSS. VSSSSS
                    DOUBLE PRECISION UR. VR. US. VS. USS. VSS. URR, VRR, UD, VD, UT, VT. EPSLON, EP
 0004
                   1SZ, EPS3, EPS4, BBH
                    DIMENSION UR(261, VR(26), US(26), VS(26), USS(26), VSS(26), URR(26), VRR(
 0005
                   126),UT(26),VT(26)
 0006
                    COMMON EPSRT, EPSLUN, EPS2, EPS3, EPS4, 102, MAX
 0007
                    N1=N
 8000
                    M1=M
 0009
                    KKK=N+1
                    DO 20 [=1.KKK
 0010
                    URR([]=UR([]
0011
 0012
                 20 VRR([]=VR([]
 0013
                    KKK=M+1
                    DO 25 I=1,KKK
USS(1)=US(1)
 0014
 0015
 0016
                 25 VSS(1)=VS(1)
                 30 BBB=USS(M1+1)*USS(M1+1)+VSS(M1+1)*VSS(M1+1)
 0017
                    UO=(URR(N1+1)+USS(ML+1)+VRR(N1+1)+VSS(M1+1))/BBB
 0018
 0019
                    VD=[USS(M1+1)*VRR(N1+1)-URR(N1+1)*VSS(M1+1))/888
 0020
                    KKK=N1+1-M1
                 DO 40 1=KKK,NI
UT(1)=URR(1)-(UD*USS(1-N1+M1)-VD*VSS(1-N1+M1))
40 VT(1)=VRR(1)-(UD*VSS(1-N1+M1)+VD*USS(1-N1+M1))
 0021
 0022
 0023
 0024
                    IF(N1.EQ.N1) GO TO 70
                    KKK=NI-M1
DO 60 (=1,KKK
UT([]=URR([]
 0025
 0026
 0027
                 60 VT(T)=VRR(1)
 0028
                 70 DO 90 [=1,N1
 0029
                    BBB=DSQRT(UT(N1+1-1)*UT(N1+1-1)+VT(N1+1-1)*VT(N1+1-1))
 0030
 0031
                    IFIBBB.GT.EPSLONI GD TO 100
 0032
                 90 CONTINUE
                    DO 95 [=1,M1
888=USS(M1+1)*USS(M1+1)+VSS(M1+1)*VSS(M1+1)
 0033
0034
 0035
                    USSSSS=[USS(1)*USS(M1+1)+VSS(1)*VSS(H1+1)1/B8B
 0036
                    VSSSS=(VSS(1)*USS(M1+1)-USS(1)*VSS(M1+1))/8B8
 0037
                    USS(11=USS5$$
                 95 VSS(1)=VSSSSS
USS(M1+1)=1.0
 0038
 0039
                    VSS(M1+1)=0.0
 0040
                    GO TO 200
0041
 0042
                100 K=N1-T
 0043
                    1F(K.EQ.01 GG TG 170
                    IF(K.LT.M1) GO TO 140
 0044
                    KKK=K+1
 0045
 0046
                    DO 130 J=1.KKK
                    URR(J)=UT(J)
 0047
 .0048
                130 VRR(J)=VT(J)
 0049
                    N1=K
```

0050		GO TO 30
0051	140	
0052		DO 150 J=1.KKK
0053		URREJI=USSEJI
0054		VRR(J)=VSS(J)
0055		USS(J)=UT(J)
0056	150	(L)TV=(L)22V
0057		KKK=K+2
0058		NNN=M1+1
0059		DO 160 J=KKK,NNI
0060		URR(J)=USS(J)
0061	160	VRR(J)=VSS(J)
0062		NI=MI
0063		M1 =K
0064		GO TO 30
0065	170	USS(1)=1.0
0066		VSS(1)=0.0
0067		M1=0
8 8 0 0 0	200	RETURN
0069		END

```
0001
                        SUBROUTINE QUADIN, UA, VA, J, UROOT, VROOT, MULTI
                ¢
                Č
                       SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR HULTIPLICITIES OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR. SOLUTION OF THE
                ¢
                        QUADRATIC IS DONE USING THE QUADRATIC FORMULA.
                Ç
                     ********
0002
                       DOUBLE PRECISION EPSRT
0003
                        DOUBLE PRECISION UA, VA, URGOT, VROOT, UDISC, VDISC, UTEMP, VTEMP, UD, VD, E
                       1PS1, EPS2, EPS4, EPSLON, BBB
                       DIMENSION UA(26), VA(26), UROOT(25), VROOT(25), MULT(25)
0004
                       COMMON EPSRT, EPS1, EPS2, EPSLON, EPS4, 102, MAX IF(N.GT.1) GO TO 60 IF(J.LT.0) GO TO 40
0005
0006
0007
0008
0009
                       GO TO 50
                    40 MULT(1)=1
0010
0011
                        3 = 1
0012
                    50 BBB=UA(21*UA(2)+VA(2)*VA(2)
0013
                       UROOT(J) == (UA(1)*UA(2)+VA(1)*VA(2))/888
0014
                        VROOT(J)=-(VA(1)*UA(2)-UA(1)*VA(2))/888
0015
                       GD TO 200
                   60 UDISC=(UA(2)*UA(2)-VA(2)*VA(2))-(4.0*(UA(3)*UA(1)-VA(3)*VA(1)))
VDISC=(2.0*UA(2)*VA(2))-(4.0*(UA(3)*VA(1)+VA(3)*UA(1)))
B8B=DSQRT(UDISC*UDISC*VDISC*VDISC)
0016
0017
0018
                       IFIBOR LE EPSLONI GO TO 100
0019
0020
                       IF(J.GE.O) GO TO 80
0021
                       MULT(1)=1
0022
                       MULT(2)=1
0023
                       J≠O
0024
                   80 CALL COMSQT(UDISC, VOISC, UTEMP, VTEMP)
0025
                       UD=2.0*UA(3)
0026
                       VD=2.0*VA(3)
0027
                       886=UD*UD+VD*VD
                       URODT(J+1)=((-UA(2)+UTEMP)*UD+(-VA(2)+VTEMP)*VD)/BBB
VRODT(J+1)=(1-VA(2)+VTEMP)*UD-(-UA(2)+UTEMP)*VD)/BBB
URODT(J+2)=((-UA(2)-UTEMP)*UD+(-VA(2)-VTEMP)*VD)/BBB
0028
0029
0030
0031
                       VROOT(J+2)=((-VA(2)-VTEMP)*UD-(-UA(2)-UTEMP)*VD)/BBB
0032
                  GO TO 200
100 IF(J.LT.0) GO'TO 110
0033
0034
0035
                       J≄J÷l
                       GO TO 130
0036
0037
                  110 MULT(1)=2
0038
                       J=1
0039
                  130 UD=2.0*UA{3}
                       VD=2.0*VA(3)
0040
                       888=UD*UD+VD*VD
0041
0042
                       UROOT(J)=(-UA(2)*UD-VA(2)*VD)/BB8
0043
                        VROOT(J)=1-VA(21*UD+UA(2)*VD)/BB8
0044
                  200 RETURN
0045
                       END
```

```
"SUBROUTENEMMULEER LUATVATNPTUAPP, VAPP, NAPP, XSTART, XEND, UROOT, VROOT, INCOT, IROOT, URAPP, VRAPP, NOPOLY)
1000
                      Ċ
                                 MULLER'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A POLYNOMIAL OF MAXIMUM DEGREE 25. THROUGH THREE GIVEN POINTS THE POLYNOMIAL IS APPROXIMATED BY A QUADRATIC. THE ZERO OF THE QUADRATIC CLOSEST TO THE OLD APPROXIMATION IS TAKEN AS THE NEW APPROXIMATION. IN THIS MANNER A SEQUENCE IS OBTAINED CONVERGING TO A ZERO.
                              ٠
                               DOUBLE PRECISION UPX3, VPX3, UPX2, VPX2, UROOT, VROOT, UX1, VX1, UAPP, VAPP
1, UX2, VX2, UWGRK, VHORK, UX3, VX3, UB, VB, UX4, VX4, UA, VA, UPX1, VPX1, URAPP, V
2RAPP, UPX4, VPX4, EPSRT, EPSO, EPS, CCC, EPSN, UH3, VH3, UQ4, VQ4, ABPX4, ABPX3
0002
0003
                                 OIMENSION UROOT (25), VROOT (25), HULT (25), UAPP(25,3), VAPP(25,3), UWORK
                                1(26), VHORK(26), UB(26), VB(26), UA(26), VA(26), URAPP(25, 3), VRAPP(25, 3)
                                 LOGICAL CONV
DOUBLE PRECISION EPS1
COMMON EPSRT, EPS1, EPS, EPS0, EPSN, 102, MAX
0004
0005
0006
0007
                                 DATA PNAME, DNAME/2HP (, 2HD (/
0008
                                 EPSRT=0.999
0009
                                 NROOT=0
0010
                                 IROOT=0
                                 [PATH#1
0011
0012
                                 NOMULT=0
                                 NALTER=0
0013
0014
                                 IT[ME=0
0015
                                 IAPP=L
                                 ITER=1
0016
                                 IF(NAPP.NE.O) GO TO 18
0017
0018
                                 NAPP=NP
0019
                                 CALL GENAPP(UAPP, VAPP, NAPP, XSTART)
                           GO TO 27
18 00 25 1=1,NAPP
UAPP([,1]=0.9*UAPP([,2]
VAPP([,1]=0.9*VAPP([,2]
0020
0021
0022
0023
0024
                                 UAPP([,3]=1.1*UAPP([,2]
0025
                                 VAPP([,3]=1,1*VAPP([,2]
                           27 KKK=NP+1
00 30 I=1+KKK
UWORK(I)=UA(I)
0026
0027
0028
                           30 VWORK(I)=VA(I)
0029
                                 NWORK=NP
0030
0031
                                 UX1=UAPP(IAPP,1)
                                 VX1=VAPP([APP.1)
UX2=UAPP([APP.2)
VX2=VAPP([APP.2)
0032
0033
0034
0035
                                 UX3=UAPP(IAPP,3)
0036
                                 VX3=VAPP([APP.3]
                           CALL HORNER(NHORK,UWORK,VWORK,UX1,VX1,UB,VB,UPX1,VPX1)
CALL HORNER(NWORK,UWORK,VWORK,UX2,VX2,UB,VB,UPX2,VPX2)
CALL HORNER(NWORK,UWORK,VWORK,UX3,VX3,UB,VB,UPX3,VPX3)
50 CALL CALCIUX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UX
14,VX4,UQ4,VQ4,UH3,VH3)
0037
0038
0039
0040
0041
                                CALL HORNER (NHORK, UWORK, VHORK, UX4, VX4, UB, VB, UPX4, VPX4)
0042
                                 ABPX4=DSQRT(UPX4+UPX4+VPX4+VPX4)
                                 ABPX3=DSQRT(UPX3+UPX3+VPX3+VPX31
0043
```

```
IFIABPX3.EQ.O.O) GO TO TO
0044
0045
                         QQQ=ABPX4/ABPX3
0046
                         IF(QQQ.LE.10.) GO TO 70
                         UQ4=0.5*UQ4
VQ4=0.5*VQ4
0047
0048
                         UX4=UX3+(UH3+UQ4-VH3+VQ4)
0049
0050
                         VX4=VX3+(VH3+UQ4+UH3+VQ4)
                    VX4=VX3+(VH3*UQ4*UH3*VQ4)
GO TO 60
TO CALL TEST(UX3,VX3,UX4,VX4,CONV)
IF(CONV) GO TO 120
IF(ITER.LT.MAX) GO TO 110
CALL ALTER(UAPP(IAPP,1),VAPP(IAPP,1),UAPP(IAPP,2),VAPP(IAPP,2),UAP
IP(IAPP,3),VAPP(IAPP,3),NALTER,ITIME)
IE(NALTER GT 5) GO TO TE
0051
0052
0053
0054
0055
                         IF(NALTER.GT.5) GO TO 75
0056
                     TTER=1
GD TO 40
75 IF(IAPP-LT.NAPPI GD TO 100
IFIXEND.EQ.O.O) GD TO 77
IF(XSTART.GT.XEND) GD TO 77
0057
0058
0059
0060
0061
0062
                        CALL GENAPP(UAPP, VAPP, NAPP, XSTART)
IAPP=0
                         NAPP=NP
0063
0064
                         GO TO 100
0065
                     77 WRITE( 102, 1090 )
0066
0067
                         KKK=NWORK+1
                         WRITE(102,1035) (DNAME,J.UWDRK(J1,VWORK(J1,J=1,KKK)
0068
                     BO IF(NROOT.EQ.O) GO TO 90
IF(IPATH.EQ.1) GO TO 82
0069
0070
0071
                     81 SPATH=2
                         CALL BETTER (UA, VA, NP, UROOT, VRQOT, NROQT, URAPP, VRAPP, EROOT, NULT)
0072
0073
                         RETURN
                     00.74
0075
0076
0077
0078
0079
                         IF(IROOT.LT.NROOT) GO TO 85
                     GO TO 87
85 KKK=IROOT+1
0080
0081
                     WRITE(102,1086) (1,URODT(1),VROOT(1),E=KKK,NRODT)
87 IF(IPATH.EQ.1) GO TO 81
0082
0083
0084
                        RETURN
0085
                     90 WRITE(102,1070) NOPOLY
0086
                         RETURN
                   100 IAPP=IAPP+L
0087
                         ITER=1
0088
0089
                         NALTER=0
                   GO TO 40
120 NRODT=NRODT+1
IROOT=NRODT
0090
0091
0092
                         MULTINACOTI=1
0093
0094
                         NOMULT=NOMULT+1
0095
                         UROOT (NROOT) =UX4
0096
                         VROOT (NROOT) = VX4
                        URAPP(NROOT,1) *UAPP(IAPP,1)
VRAPP(NROOT,1) = VAPP(IAPP,1)
URAPP(NROOT,2) = UAPP(IAPP,2)
0097
0098
0099
```

VRAPP(NROOT, 2) = VAPP((APP, 2)

```
URAPP(NROOT;3)=UAPP(IAPP;3)
VRAPP(NROOT;3)*VAPP(IAPP;3)
 0101
 0102
 0103
                       125 IF(NOMULT.LT.NP) GO TO 130
 0104
                             GO TO 80
                            CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,V8,UPX4,VPX4)
NWORK=NWORK-1
0105
 0106
0107
                             KKK=NWORK+1
                             DO 140 [=1,KKK
UWDRK([]=UB(])
0108
 01 09
0110
                       140 VWORK(I)=VB(I)
                             CALL HORNER(NWORK, UWORK, UWA, VX4, UB, VB, UPX4, VPX4)
CCC=DSQRT(UPX4*UPX4+VPX4*VPX4)
IF(CCC.LT.EPSM) GO TO 150
0111
0113
0114
                             IFINWORK.GT.2) GO TO 75
0115
                             I RODT=NROOT
0116
                             KKK=NWORK+1
                      DD 145 [=],KKK
UB(1)=UWORK(KKK+1-1)
145 VB(1)=VWORK(KKK+1-1)
0117
8110
0119
0120
                             CALL QUADINWORK, UB, VB, NROOT, UROOT, VROOT, MULTI
0121
                             GO TO 80
0122
                      150 MULT(NROOT)=MULT(NROOT)+1
                      NOMULT=NOMULT+1
GO TO 125
L10 UX1=UX2
0123
0124
0125
0126
                             VX1=VX2
0127
                             UX2=UX3
0128
                             VK2=VX3
0129
                             UX3=UX4
0130
0131
                             VX3=VX4
                             UPX1=UPX2
0132
                             VPX1=VPX2
0133
                             UPX2=UPX3
                             VPX2=VPX3
0134
                            UPX3=UPX4
VPX3=VPX4
0135
0136
0137
                             ITER=ITER+1
                     GO TO 50
1090 FORMAY1///.1x.65HCDEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO
0138
0139
                     12EROS WERE FOUND//1
1080 FORMAT(///1x,13HROOTS OF Q(x),83x,21HINITIAL APPROXIMATION//)
1070 FORMAT(//,43H NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER .[2]
1086 FORMAT(2X,5HROOT(,12,4H) = ,023,16,3H + ,D23,16,2H I,19x,23HSQLVED 1 BY DIRECT METHOD)
0140
0141
0142
                     1035 FORMAT(3x,A2,I2,4H) = ,D23.16,3H + ,D23.16,2H I)
1050 FORMAT(82x,D23.16,3H + ,D23.16,2H I/82x,D23.16,3H + ,D23.16,2H I/1
1085 FORMAT(2x,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,18x,D23.16,3H
0143
0144
0145
                           1 + ,023.16,2H II
0146
```

```
SUBROUTINE BETTER (UA, VA, NP, UROOT, VROOT, NROOT, URAPP, VRAPP, TROOT, MUL
1000
                          SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND BY USING THEM AS INITIAL APPROXIMATIONS WITH MULLER'S METHOD APPLIED TO THE FULL, UNDEFLATED POLYNOMIAL.
0002
                          DOUBLE PRECISION URDOT. VROOT, UA, VA, UBAPP, VBAPP, UX1, VX1, UX2, VX2, UX3
                        1.VX3.UPX1.VPX1.UPX2.VPX2.UPX3.VPX3.UB.VB.URGOTS.VRGOTS.EPSRT.UX4.V
                        2X4,URAPP, VRAPP, EPSO, EPS, UQ4, VQ4, UH3, VH3
                          LOGICAL CONV
0003
                          DIMENSION UROOT (25), VROOT (25), UA(26), VA(26), UBAPP(25,3), VBAPP(25,3
0004
                        1). UB1261, VB(261, UROOTS(25), VROOTS(25), URAPP(25,3), VRAPP(25,3), MULT
0005
                          DOUBLE PRECISION EPS1.EPSM
                          COMMON EPSRT, EPS1, EPS, EPSO, EPSM, [O2, MAX IF(NROOT, LE. 1) RETURN
0006
0007
0008
                          L=0
0009
                          DO 10 I=1.NROOT
0010
                          UBAPP(I,1)=URGOT(I)*EP$RT
0011
                          VBAPP([.1]=VROOT([]+EP$RT
                          UBAPP(1,2)=UROOT([)
0012
                         VBAPP(1,2)=VROOT(1)
UBAPP(1,3)=UROOT(1)*(2.0-EPSRT)
VBAPP(1,3)=VROOT(1)*(2.0-EPSRT)
0013
0014
0015
                          DO 100 J=1,NROOT
UX1=U6APP(J,1)
VX1=VBAPP(J,1)
0016
0017
0018
                          UX2=UBAPP(J,2)
0019
                          VXZ=VBAPP(J,2)
0020
0021
                          UX3=UBAPP(J,31
0022
                          (E, L) 99ABV=EXV
0023
                          ITER≈1
                         CALL HORNER(NP,UA,VA,UX1,VX1,UB,VB,UPX1,VPX1)
CALL HORNER(NP,UA,VA,UX2,VX2,UB,VB,UPX2,VPX2)
CALL HORNER(NP,UA,VA,UX3,VX3,UB,VB,UPX3,VPX3)
0024
0025
0026
0027
                          CALL CALCIUX1, VX1, UX2, VX2, UX3, VX3, UPX1, VPX1, UPX2, VPX2, UPX3, VPX3, UX
                     14, VX4, UQ4, VQ4, UH3, VH3)
30 CALL TESTIUX3, VX3, UX4, VX4, CONV)
IFICONV) GQ TQ 50
0028
0029
                          [F(ITER.LT.MAX) GD TD 40
WRITE(102,1000) J.URODT(J), VRODT(J), MAX
WRITE(102,1010) UX4,VX4
0030
0031
0032
                         IF(J.LT.IROOT) GO TO 33
IF(J.EQ.IROOT) GO TO 35
GO TO 100
0033
0034
0035
                     33 KKK=IRODY-L
0036
0037
                          00 34 K=J,KKK
                          URAPP(K,1)=URAPP(K+1,1)
0038
                          VRAPP(K.1)=VRAPP(K+1.1)
0039
                         URAPP(K,2)=URAPP(K+1,2)
VRAPP(K,2)=VRAPP(K+1,2)
0040
0041
                          URAPP(K, 3)=URAPP(K+1,3)
0042
0043
                          VRAPP(K,3)=VRAPP(K+1,3)
0044
                     35 IROOT=IROOT-1
0045
                         GO TO 100
```

```
0046
0047
0048
0049
                             40 UX1=UX2
VX1=VX2
UX2=UX3
VX2=VX3
0050
                                   UX3=UX4
0051
                                   VX3=VX4
                                   UPX1=UPX2
VPX1=VPX2
UPX2=UPX3
VPX2=VPX3
VPX2=VPX3
VPXE=VPX3
0052
0053
0054
0055
0056
                           GO TO 20

50 L=1+1

URODTS(1)=UX4

VROOTS(1)=VX4

100 CDNTINUE
0057
0058
0059
0060
0061
                           If(L.EQ.0) GO TO 120

DO 110 [=1;4

UROOT(1)=UROOTS(1)

110 VRDDY(1)=VROOTS(1)

NROOT=1
0062
0063
0064
0065
0066
0067
                                   RETURN
                           120 NR00T=0
8 400
0069
                                   RETURN
                         1000 FORMAT(///42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,12,4H) = , 1023.16,3H + ,023.16,2H 1/24H DIO NOT CONVERGE AFTER ,13,11H TTERAT 210NS)
0070
0071
                          1010 FORMATISON THE PRESENT APPROXIMATION IS ,D23.16,3H + ,023.16,2H 1/
                                 L/)
0072
```

```
SUBROUTINE ALTERIXIR . X11 . X2R . X2I . X3R . X3I . NALTER . I TIME )
0001
                  20000
                          SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT.
                           DOUBLE PRECISION X1R.X11.X2R.X21.X3R.X31.EP$1.EP$2.EP$3.R.BETA
0002
                           DOUBLE PRECISION EPS4.EPS5
COMMON EPS1,EPS2,EPS3,EPS4,EPS5,102,MAX
IF(ITIME.NE.O) GO TO 5
0003
0004
0005
                           ITIME=1
0006
                        WRITE(102.1010) MAX
5 IF(NALTER.EQ.0) GO TO 10
WRITE(102.1000) XIR.XII.X2R.X2I.X3R.X3I
0007
000B
0009
                      GO TO 20
10 R=DSQRT(X2R*X2R+X21*X2I)
0010
0011
0012
                           BETA=DATAN2(X21, X2R)
0013
                           WRITE(102,1020) X1R,X11,X2R,X21,X3R,X31
                      20 NALTER=NALTER+1
1F(NALTER.GT.S) RETURN
GO TO (30,40,30,40,30),MALTER
0014
0015
0016
0017
                      30 X2R=-X2R
0018
                           X21=-X21
                      GO TO 50
40 BETA=BETA+1.0471976
0019
0020
                      X2R=R*OCOS(BETA)
X2I=R*OSIN(BETA)
50 X1R=0.9*X2R
0051
0022
0023
0024
                           X11=0.9*X21
0025
                           X3R=1.1*X2R
9500
                           X3I=1.1*X2I
                           RETURN
0027
                   1000 FORMAT(1X,5HX1 = .D23.16.3H + .D23.16.2H 1.10X,22HALTERED APPROXIN
1AT(DNS/1X,5HX2 = .D23.16.3H + .D23.16.2H 1/1X,5HX3 = .D23.16.3H +
0028
                          2.023.16.24 [/]
                   1020 FORMAT(1H0,5HXL = .D23.16.3H + .D23.16.2H I.10X,22HINITIAL APPROXI

1MAT10NS/1X,5HX2 = .D23.16.3H + .D23.16.2H I/1X,5HX3 = .D23.16.3H +
0029
                   2 .D23.16.2H [/]
1010 FORMAT(///1x,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
0030
                         ITER ,13,12H ITERATIONS.//I
0031
                           END
```

```
0001
                             SUBROUTINE GENAPPIAPPR, APPI, NAPP, XSTARTS
                   000000
                            SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE DEGREE OF THE ORIGINAL POLYNOMIAL.
                            DOUBLE PRECISION APPR.APPI.xSTART.EPS1.EPS2.EPS3.BETA
DOUBLE PRECISION EPSRT.EPS4
DIMENSION APPR(25,3).APPI(25,3)
COMMON EPSRT.EPS1.EPS2.EPS3.EPS4.IO2.MAX
IF(XSTART.EQ.0.0) XSTART=0.5
0002
0003
0004
0005
0006
                       0007
0008
0010
1100
0012
0013
0014
0015
9100
0017
0018
                            RETURN
0019
                            END
```

```
SUBROUTINE TESTIUX3 .VX3.UX4.VX4.CONV)
1000
                           SUBROUTINE TEST CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROXIMATIONS BY TESTING THE EXPRESSION
ABSOLUTE VALUE OF (X(N+1)-X(N))/ABSOLUTE VALUE OF X(N+1).
WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.
                           DOUBLE PRECISION UX3, VX3, UX4, VX4, EPSRT, EPSO, EPS, AAA, UDUMNY, VDUMNY,
0002
                          1DENOM
                           LOGICAL CONV
0003
                           DOUBLE PRECISION EPSI.EPSM
COMMON EPSRT.EPSI.EPS.EPSG.EPSM.IOZ.MAX
0004
0005
0006
                            UDUMMY=UX4-UX3
                           VDUHNY=VX4-VX3
AAA=DSQRT(UDUHNY+UDUMNY+VDUMNY+VDUMNY)
0007
0008
0009
                           DENON-DSQRT(UX4+UX4+VX4*VX4)
                           IF(DENOM.LT.EPSO) GO TO 20
IF(AAA/DENOM.LT.EPS) GO TO 10
0010
0011
                       5 CONV=.FALSE.
GD TO 100
10 CONV=.TRUE.
0012
0013
0014
0015
                           GD TO 100
0016
                          IFIAAA.LT.EPSO) GO TO 10
0017
                           GO TO 5
0018
                     100 RETURN
0019
                           ENG
```

```
0001
                             SUBROUTINE HORNER(NA, UA, VA, UX, VX, UB, VB, UPX, VPX)
                            HORNER'S METHOD COMPUTES THE VALUE OF THE POLYMONIAL P(X) AT A POINT O. SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYMONIAL BY DIVIDING OUT THE
                  C
C
                            FACTOR (X-D).
                  C
                            DOUBLE PRECISION UX, VX, UPX, VPX, UB, VB, UA, VA
DIMENSION UA(26), VA(26), UB(26), VB(26)
0002
0003
0004
                            UB(1)=UA(1)
0005
                            VB(1)=VA(1)
0006
                            NUM=NA+1
                            00 10 7=2, NUM
UB(1)=UA(1)+(UB(1-1)*UX-VB(1-1)*VX)
VB(1)=VA(1)+(VB(1-1)*UX+UB(1-1)*VX)
0007
0008
0009
                             (MUM) BU=X9U
0010
                             VPX=VB(NUM)
0011
0012
                            RETURN
0013
                            END
```

```
0001
                             SUBROUTINE CALCIUX1, VXI, UXZ, VXZ, UX3, VX3, UPX1, VPX1, UPX2, VPX2, UPX3, V
                    C
                    C
                          * GIVEN THREE APPROXIMATIONS X(N-2), X(N-1), AND X(N), SUBROUTINE CALC * APPROXIMATES THE POLYNOMIAL BY A QUADRATIC AND SOLVES FOR THE ZERO OF THE QUADRATIC CLOSEST TO X(N). THIS ZERO IS THE NEW APPROXIMATION * X(N+1) TO THE ZERO OF THE POLYNOMIAL.
                   ۲
0
0
                           1PX3,UX4,VX4,UQ4,VQ4,UH3,VH3)

DOUBLE PRECISION ARGI,ARG2

DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UX1,VX1,UX2,VX2,UX3,VX3,UPX1,
1VPX1,UH3,VH3,UH2,VH2,UQ3,VQ3,UD,VD,UB,VB,UC,VC,UDISC,VDISC,UCCC,VC
2CC,UDEN1,VDEN1,UDEN2,VDEN2,UD4,VQ4,UX4,VX4,EPSRT,EPSO,EPS,UDDD,VDD
 0002
 0003
                            3D, AAA, BBB, RAD, UAAA, VAAA, UBBB, VBBB
                            DOUBLE PRECISION THETA, ANGLE, UTEST, VTEST
DOUBLE PRECISION EPSI, EPSM
COMMON EPSRT, EPSI, EPS, EPSM, EPSM, 102, MAX
 0004
 0005
 0006
 0007
                             UH3≈UX3~UX2
 0008
                             VH3=VX3-VX2
                             UH2=UX2-UX1
VH2=VX2-VX1
BBB=UH2*UH2+VH2*VH2
 0009
 0010
 0011
 0012
                             UQ3=(UH3*UH2+VH3*VH2)/888
 0013
                             VQ3=(VH3*UH2-UH3*VH2)/888
 0014
                             UDD0=1.0+UQ3
 0015
                             VDDD=VQ3
                             UD=(UPX3-(UDDD+UPX2-VDDD+VPX2))+(UQ3*UPX[-VQ3+VPX])
VD=(VPX3-(VDDD+UPX2+UDDD+VPX2))+(VQ3+UPX1+UQ3+VPX])
 0016
 0017
 0018
                             EQU+0.S=AAAU
 0019
                             VAAA=2.0+VQ3
 0020
                             O.I+AAAU=AAAU
0021
                             000V*000V-000U+000
                             V888=VD0D+UDDD+UDDD+VDDD
0022
 0023
                             UCCC=UQ3+UQ3-VQ3+VQ3
0024
                             VCCC=VQ3+UQ3+UQ3+VQ3
0025
                            UB= { { UAAA*UPX3-VAAA*VPX3}-{ UBBB*UPX2-VBBB*VPX2} } + { UCCC*UPX1-VCCC*V
                           1 P X L 1
9500
                             VB=((VAAA+UPX3+UAAA+VPX31-(VBB6+UPX2+UBB8+VPX2))+(VCCC+UPX1+UCCC+V
                           [PX1]
 0027
                            UC=UDDD*UPX3~VDDD*VPX3
0028
                             VC=VDDD+UPX3+UDDD+VPX3
0029
                            UDISC=[UB*UB~VB*VB]~[4.0*[UD*UC-VO*VC]]
                            VD1 SC = t2.0*(VB*UB)}-(4.0*(VD*UC+UD+VC)}
AAA=DSQRT(UD1SC*UD1SC+VD1SC+VD1SC)
0030
0031
0032
                            IF(AAA.EQ.O.Q) GO TO 5
0033
                            GO TO 7
0034
                            THETA=0.0
                           GO TO 9
THETA=DATAN2(VDISC, UDISC)
RAD=DSQRT(AAA)
ANGLE*THETA/2.0
0035
0036
0037
0038
0039
                            UTEST=RAD+DCOS(ANGLE)
0040
                            VTEST=RAD+DS IN (ANGLE)
0041
                            UDEN1=UB+UTEST
                            VDEN1=VB+VTEST
                            UDEN2=UB-UTEST
0043
0044
                            VDEN2=V8-VTEST
```

<del></del> .		
0045		ARG1=UDEN1*UDEN1+VDEN1*VDENL
0046		ARGZ=UDEN2+UDEN2+VDEN2+VDEN2
0047		AAA=OSORT (ARG1)
0048		BBB=DSQRT (ARG2)
0049		IF(AAA.LT.BBB) GO TO 10
0050		IF(AAA.EQ.O.D) GD TO 60
0051		UAAA=-2.0+UC
0052	•	VAAA=-2.0*VC
0053		UQ4={UAAA*UDEN1+VAAA*VDEN1}/ARG1
0054		VQ4=(VAAA*UDEN1-UAAA*VDEN1)/ARG1
0055		GO TO 50
0056		IF(888.EQ.O.O) GO TO 60
0057		UAAA=-2.0*UC
0058		VAAA=-2.0*VC
0059		UQ4=[UAAA+UDEN2+VAAA+VDEN2]/ARG2
0060		VQ4=(VAAA+UDEN2-UAAA+VDEN2)/ARG2
0061		GQ TO 50
9062	50	UX4=UX3+(UH3+UQ4-VH3+VQ4)
0063		VX4=VX3+{VH3+UQ4+UH3+VQ4}
0064		RETURN
0065	60	UQ4=1.0
0066	-	VQ4=0.0
0067		GO TO 50
0068		END
4555		

```
....0001
                                                "SUBROUTINE CONSQUIUX, VX, UY, VY)
                                    00000
                                               * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.
                                           DOUBLE PRECISION UK, VX.UY, VY.DUMMY.R.AAA.BBB
R=DSQRT(UX*UX+VX*YX)
AAA=DSQRT(DABS((R-UX)/2.0])
BBB=DSQRT(DABS((R-UX)/2.0])
IF(VX) 10.20.30

10 UY=AAA
VY=-1.0*BBB
GD TO 100
20 IF(UX) 40.50.60
30 UY=AAA
VY=BBB
GO TO 100
40 DUMMY=DABS(UX)
UY=0.0
      0003
     0004
0005
0006
      0007
      8000
      0009
     0010
0011
0012
0013
     0014
                                                   0.0=YÜ
                                        UY=0.0
VY=DSQRT(DUMMY)
GO TO 100
SO UY=0.0
VY=0.0
GO TO LOO
60 DUMMY=DABS(UX)
UY=DSQRT(DUMMY)
VY=0.0
100 RETURN
END
     0016
0017
0018
0019
     0020
     0021
0022
0023
0024
     0025
```

#### APPENDIX G

#### REPEATED G.C.D. - NEWTON'S METHOD

#### 1. Use of the Program

A double precision FORTRAN IV program using the repeated G.C.D. method with Newton's method as a supporting method is presented here. Flow charts for this program are given in Figure G.2 while Table G.III gives a FORTRAN IV listing of this program. Single precision variables are listed in Table G.II. The single precision variables are used in the flow charts and the corresponding double precision variables can be obtained from Table G.II.

This program is designed to solve polynomials having degree less than or equal to 25. In order to solve polynomials of degree N where N > 25, the data statement and array dimensions given in Table G.I. must be changed.

In this program both the leading coefficient and the constant coefficient are assumed to be non-zero.

#### TABLE G.I

PROGRAM CHANGES NECESSARY TO SOLVE POLYNOMIALS OF DEGREE GREATER THAN 25 BY THE REPEATED G.C.D. - NEWTON'S METHOD

#### Main Program

Data Entry/1H1,1H2,...,1H9,2H10,2H11,...,2HXX/where XX = N+1 UP(N+1), VP(N+1)UAPP(N), VAPP(N) UDO(N+1), VDO(N+1) UDDO(N+1), VDDO(N+1) UD1(N+1), VD1(N+1) UD2(N+1), VD2(N+1)UDD1(N+1), VDD1(N+1) UG(N+1), VG(N+1) UD3(2N+1), VD3(2N+1)UD4(2N+1), VD4(2N+1)UZROS(N), VZROS(N) UAP(N), VAP(N) UROOT (N), VROOT (N) NULT (N) ENTRY (N+1)

#### Subroutine PROD

UH(2N+1), VH(2N+1) UF(N+1), VF(N+1) UG(N+1), VG(N+1)

#### Subroutine ZROS

UAPP(N), VAPP(N)
UROOT(N), VROOT(N)
UQ(N+1), VQ(N+1)
UQQ(N+1), VQQ(N+1)
UAP(N), VAP(N)
UQD(N+1), VQD(N+1)
ENTRY(N+1)
UROOTS(N), VROOTS(N)

Subroutines GENAPP, GCD, NEWTON, DIVIDE, HORNER, and DERIV

See corresponding subroutine in Table E.I.

#### Subroutine QUAD

UROOT(N), VROOT(N) UA(N+1), VA(N+1)

#### 2. Input Data for Repeated G.C.D. - Newton's Method

The input data for repeated G.C.D. - Newton's method is prepared as described for G.C.D. - Newton's method in Appendix E, § 2 except that the item EPS4 on the control card (Figure E.2) is omitted. An example control card for the repeated G.C.D. - Newton's method is given in Figure G.1.

#### 3. Variables Used in Repeated G.C.D. - Newton's Method

The definitions of variables used in repeated G.C.D. - Newton's method are given in Table G.II. For definitions of variables not listed in this table, see the main program or corresponding subprogram of Table E.VI. The notation and symbols used are defined in Appendix E, § 3.

#### Description of Program Output

The number of the polynomial, control data, degree and coefficients of the polynomial are printed as described in Appendix E, § 4.

All roots of multiplicity one are extracted first. Following the first row of asterixes, the message "THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY 1." This is followed by the coefficients of G(X) with the leading coefficient listed first. If there are no roots of multiplicity one, then the message "NO ROOTS OF MULTIPLICITY ONE" is printed.

The roots of G(X) are printed under the heading "ROOTS OF G(X)." These are the roots obtained before the attempt to improve accuracy. The initial approximations producing convergence to the corresponding root are printed under the heading "INITIAL APPROXIMATION." The

message "RESULTS OF SUBROUTINE QUAD" means that the corresponding root was obtained from subroutine QUAD.

The roots found as a result of attempting to improve accuracy are printed under the heading "ROOTS OF P(X)." Their multiplicity is given under the heading "MULTIPLICITIES." The initial approximation is printed above where "NO INITIAL APPROXIMATION" means the same as "RESULTS OF SUBROUTINE QUAD."

A line of asterixes is then printed. This procedure is then repeated for the roots of multiplicity 2,3,4, etc. until all roots have been found.

#### 5. Informative Messages and Error Messages

The informative messages and error messages for repeated G.C.D. - Newton's method are given below. For those not listed, see Appendix E, § 5.

"NOT ALL ROOTS OF THE ABOVE POLYNOMIAL, G, WERE FOUND." This message indicates that some of the roots of the polynomial G(X) were not extracted.

"QUAD FOUND XXX TO BE A MULTIPLE ROOT." XXX represents the value of the root found as a multiple root by Subroutine QUAD.

	0000 3456	1 4			•						44444445 34567890				7777777 12345678	7 7 8 39 0
N O P	N P	N A P		MAX	EPS	1	EPS2	2	EPS	3		-		XSTART	XEND	C H
L		P						***************************************		-						CK
1	11	7		200	1.D-	03	1.D-J	LO	L.D-	20		···········		1.0D+01	2.0D+01	Ш

Figure G.1 Control Card for Repeated G.C.D. - Newton's Method

TABLE G.II

REPEATED GCD - NEWTON'S METHOD

Single Pre	cision	Double Prec	ision	Disposition	
Variable	Туре	Variable	Type	of Argument	Description
•				Мэ	in Program
		•		rica	m itogiam
KD	I	KD	I		Number of distinct roots found
K	I	K	I		Number of roots found
Jl	I	J1	I		Multiplicity of given root
DO	C	UDO, VDO	D		Array of coefficients of original polynomial
NDO	I	ND0	I		Degree of original polynomial
DDO	С	ODDO, VDDO	D		Array of coefficients of derivative of DO(X) i.e. DO'(X)
NDDO	I	NDD0	I.	•	Degree of DDO(X)
DI	C	UD1,VD1	D		Array of coefficients of g.c.d. of DO(X) and DDO(X)
ND1	I	NDI	I		Degree of D1(X)
DD1	C	UDD1,VDD1	. D		Array of coefficients of derivative of D1(X) i.e. D1'(X)
NDDI	I	NDD1	I		Degree of DD1(X)
D2	С	UD2,VD2	D		Array of coefficients of g.c.d. of D1(X) and DD1(X)
ND2	I	ND2	I		Degree of D2(X)
. D3	С	UD3,VD3	D		Array of coefficients of the product of DO(X) and D2(X)
ND3	I	ND3	I	·	Degree of D3(X)
D4	· C	UD4,VD4	D		Array of coefficients of the square of D1(X)
ND4	I	ND4	I		Degree of D4(X)
G ·	C	UG,VG	Ď		Array of coefficients of the quotient $D3(X)/D4(X)$
NG	I	NG	I	•	Degree of G(X)
ZROS	C	UZROS, VZROS	a		Array of roots of G(X)
•	-			Sub	routine ZROS
:		· 		_	
APROX	С	UAPROX, VAPRO	)X·D	R	Starting approximation (initial or altered)
		· · · · · · · · · · · · · · · · · · ·			

TABLE G.II (Continued)

Single Pre Variable	Cision Type	Double Pre Variable	cision Type	Disposition of Argument	Description
				Subi	coutine PROD
М	I	M	I	E	Degree of polynomial to be multiplied
F	C	UF, VF	D	E ·	Array of coefficients of polynomial to be multiplied
N	I	N	I	E	Degree of polynomial to be multiplied
G	- C	UG, VG	D	Έ	Array of coefficients of polynomial to be multiplied
MN	I	MN	I	R	Degree of product polynomial H(X)
H	С	UH, VH	D	R	Array of coefficients of product polynomial
LIMIT	I	LIMIT	I		Number of coefficients of polynomial F(X)
K	I	K	I		Counter

### MAIN PROGRAM

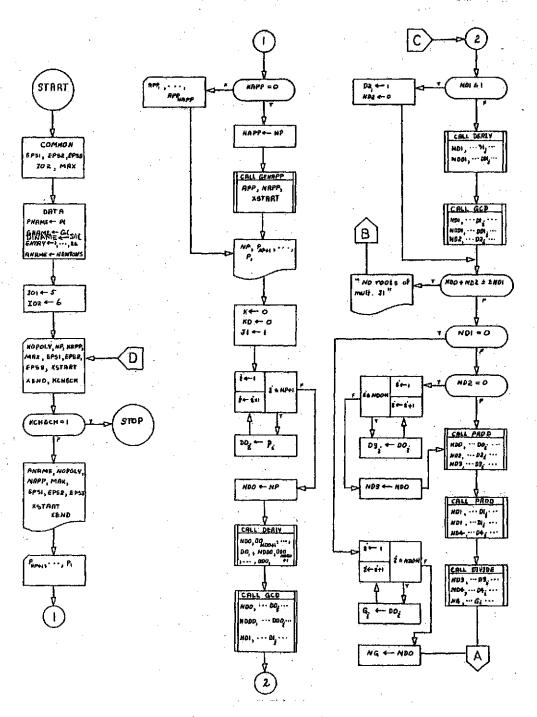


Figure G.2. Flow Charts for Repeated G.C.D.-Newton's Method

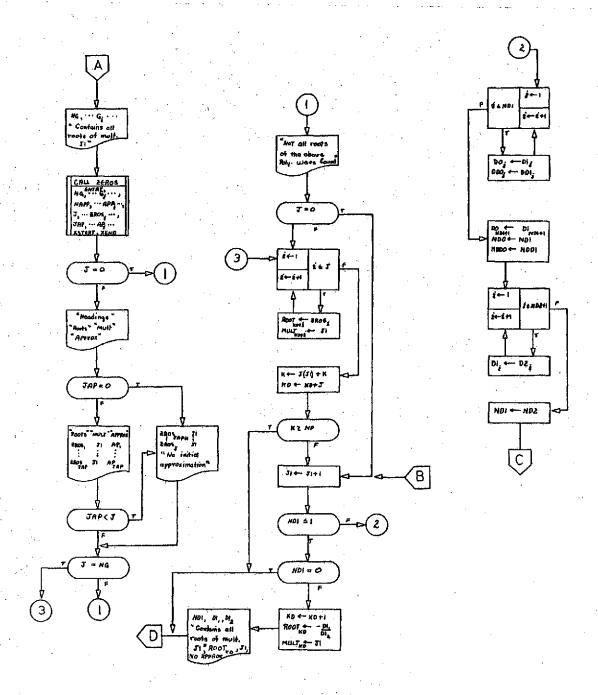


Figure G.2. (Continued)

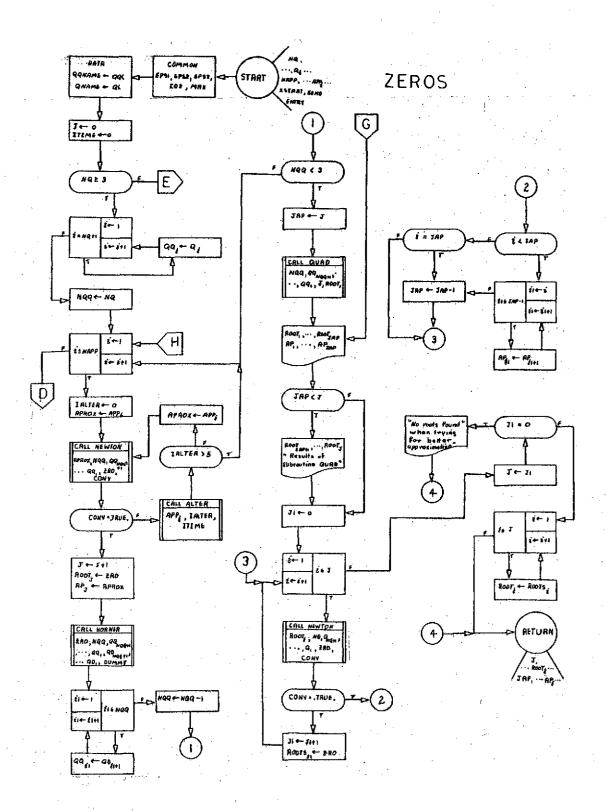


Figure G.2. (Continued)

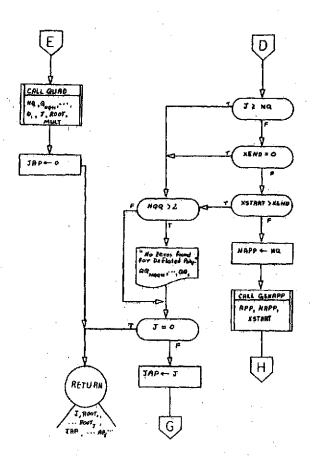


Figure G.2. (Continued)

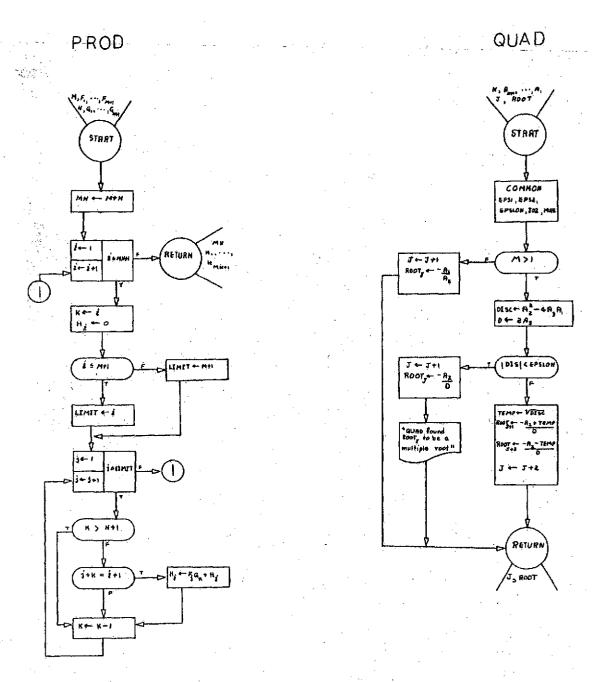


Figure G.2. (Continued)

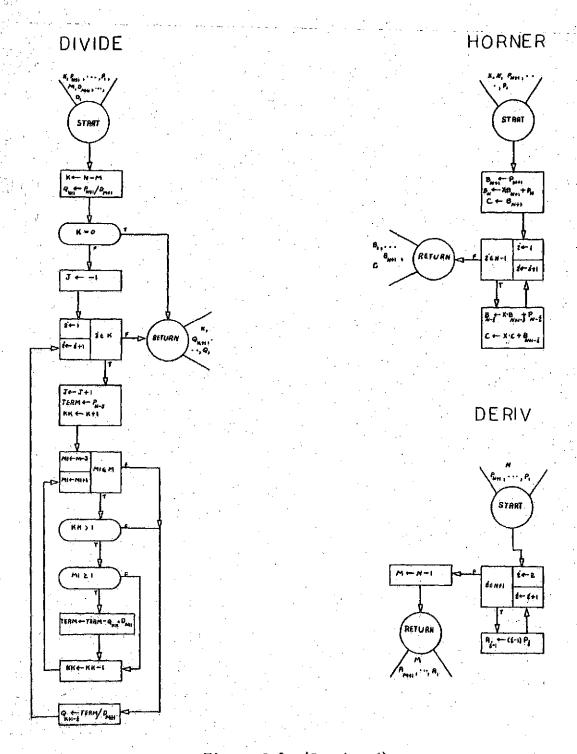
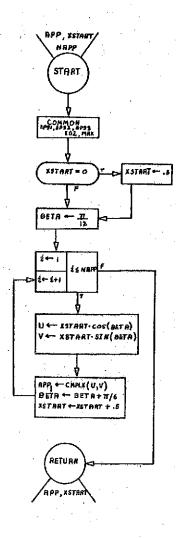


Figure G.2. (Continued)

# GENAPP

### **FILTER**



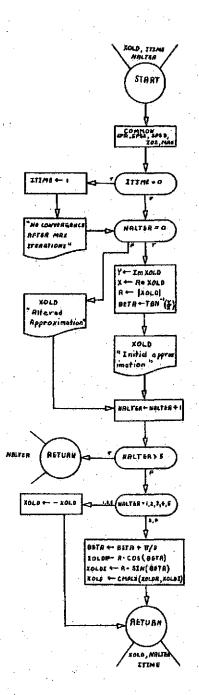


Figure G.2. (Continued)

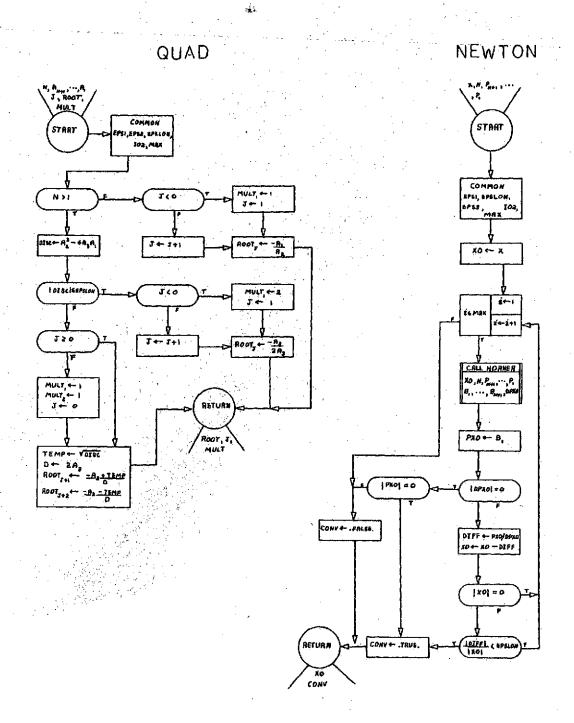


Figure G.2. (Continued)

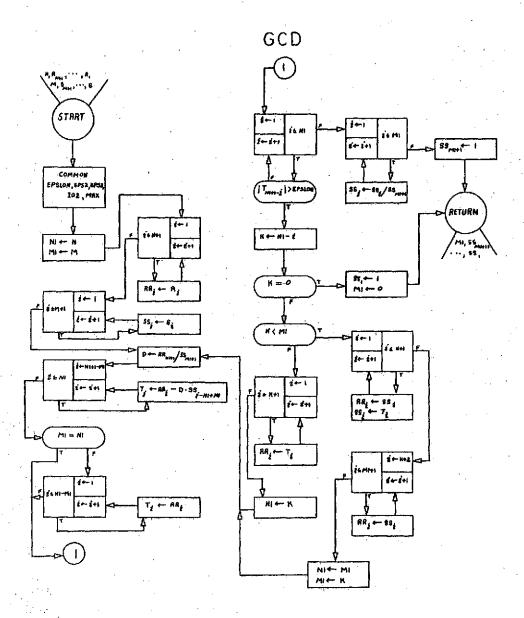


Figure G.2. (Continued)

# COMSQT

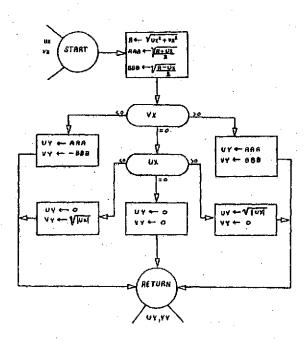


Figure G.2. (Continued)

TABLE G.III

#### PROGRAM FOR REPEATED G.C.D.-NEWION'S METHOD

```
DOUBLE PRECISION PROGRAM FOR THE REPEATED G.C.D. - NEWTON'S METHOD
                              C
                              C
                                               THIS METHOD REPEATEDLY FINDS THE GREATEST COMMON DIVISOR OF TWO
                                              POLYNOMIALS IN ORDER TO EXTRACT THE ZEROS IN GROUPS ACCORDING TO MULTIPLICITY USING NEWTON'S METHOD. ALL ZEROS OF MULTIPLICITY I ARE EXTRACTED FOLLOWED BY THOSE OF MULTIPLICITY 2, ETC.
                              C
                                               DOUBLE PRECISION EPS1.EPS2.EPS3.UP.VP.UAPP.VAPP.UD0.VD0.UDD0.VD00.
0001
                                             1UD1, VD1, UD2, VD2, UDD1, VDD1, UG, VG, UD3, VD3, UD4, VD4, UZROS, VZROS, UAP, VA
                                            2P.URODT.VROOT.DENDM
OOUBLE PRECISION XSTART
OOUBLE PRECISION XEND
0002
0003
                                               DIMENSION ANAME(2), UP(26), VP(26), UAPP(25), VAPP(25), UDO(26), VDO(26)
0004
                                            L,UDDO(26),VDD(26),UD1(26),VD1(26),UD2(26),UDD(26),UDD(26),VDD(26),UDD(26),VDD(26),UDD(26),UD3(51),UD4(51),VD4(51),UZROS(25),VZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),UZROS(25),U
0005
0006
                                              DATA PNAME,GNAME/2HP(,2HG(/, DINAME/3HD) (/
DATA ENTRY/1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,2H10,2H11,2H12,2H13
0007
0008
                                             1,2H14,2H15,2H16,2H17,2H18,2H19,2H20,2H21,2H22,2H23,2H24,2H25,2H26/
                                               DATA ANAMELLI, ANAMELZ) /4 HNEWT, 4 HONS /
0009
                                               101=5
0010
                                               102=6
00.11
                                          1 READ(101,1000) NUPOLY, NP, NAPP, MAX, EPS1, EPS2, EPS3, XSTART, XEND, KCHEC
0012
                                            1K
                                               IF(KCHECK.EQ.1) STOP
WRITE([02,1020] ANAME(1).ANAME(2).NOPOLY
0013
0014
                                               WRITE(102,2000) NAPP
0015
0016
                                               WRITE(102,2010) HAX
                                               WRITE(102,2070) EP51
0017
0018
                                               WRITE(102,2020) EP$2
                                               WRITE(102,2080) EPS3
0019
                                               WRITE(102,2040) XSTART
0020
                                               WRITELLOZ.20501 XEND
0021
                                               WRITE(102,2060)
0022
0023
                                               KKK=NP+1
0024
                                               NNN=KKK+1
0025
                                               00 5 I=1.KKK
                                               1-NNN=LLL
0026
                                         5 READITO1, 1010) UP(JJJ), VP(JJJ)
0027
                                               IF (NAPP. NE. O) GO TO 22
0028
0029
                                               NAPP=NP
0030
                                               CALL GENAPPIUAPP, VAPP, NAPP, XSTART)
0031
                                               GD TO 23
                                       22 READ(101,1015) (UAPP(11,VAPP(1),1=1,NAPP)
0032
                                       23 WRITE(102,1030) NP
0033
0034
                                               KKK=NP+1
0035
                                               NNN=KKK+1
0036
                                               DO 8 [=],KKK
                                               1-MMM=LLL
0037
                                          8 WRITE(102,1040) PMAME, ENTRY(JJJ), UP(JJJ), VP[JJJ]
0038
                                               K =0
0039
0040
                                               KD=0
```

```
0041
                           Ĵ1=1
0042
                           KKK=NP+1
                           00 10 1=1,KKK
UD0(1)=UP(1)
0043
0044
                      10 V00(11=VP(1)
ND0=NP
0045
0046
                           CALL DERIVINGO. UDO. VDO. NDOO. UDUO. VDDO!
0047
                      CALL GCD(NDO, UDO, YDO, NDDO, UDDO, VDOO, ND1, UD1, VD1)
20 WRITE(102,3000) (ASTER, I=1,33)
0048
0049
0050
                           IF(ND1.LE.11 GO TO 30
                      GO TO 40
30 UD2(11=1.0
0051
0052
0053
                           VD2(1)=0.0
0054
                           N02=0
0055
                           GO TO 50
                      40 CALL DERIVINDI, UDI, VDI, NDDI, UDDI, VDDI)
CALL GCO(NDI, UDI, VDI, NDDI, UDDI, VDDI, NDZ, UDZ, VDZ)
50 IF(NDO+NDZ, LE, 2*NDI) GO TO 60
0056
0057
0058
                      GO TO 70
60 WRITE(102,1025) J1
0059
0060
                      GO TO 170
70 [F(ND1.EQ.O) GO TO 80
1 800
0062
                      80 KKK=NDQ+1
0063
0064
                          DO 85 1=1.KKK
UG(1)=UDO(1)
0065
0066
0067
                      85 VG([]=V00([)
                          NG=NDO
GO TO 110
0068
0069
                     90 IF(ND2.EQ.0) GO TO 115
CALL PROD(ND0,UD0,VD0,ND2,UD2,VD2,ND3,UD3,VD3)
100 CALL PROD(ND1,UD1,VD1,ND1,UD1,VD1,ND4,UD4,VD4)
0070
0071
9072
                     CALL DIVIDE(ND3,UD3,VD3,ND4,UD4,VD4,NG,UG,VG)
110 WAITE(102,1035) J1
0073
0074
0075
                           KKK=NG+1
0076
                          NNN=KKK+1
0077
                           DO 115 I=1*KKK
8700
                           1-MMM=LLL
                    112 WRITE(102,1040) GNAME, ENTRY(JJJ), UG[JJJ), VG[JJJ)
CALL ZEROSING, UG, VG, NAPP, UAPP, VAPP, J, UZROS, VZROS, JAP, UAP, VAP, ENTRY
1, XSTART, XEND)
0079
0080
0081
                           IF(J.EQ.0) GO TO 150
0082
                           WRITE([02,1180]
                    IF(JAP.EQ.0) GO TO 120
GO TO 130
115 KKK=NDO+1
0083
0084
0085
                          DO 116 [=1.KKK
0086
0087
                           UD3(1)=UD0(1)
                          V03([]=V00([]
8800
                          NO3=ND0
GO TO 100
0089
0090
                    120 KKK=JAP+L
0091
                          WRITE(102,1085) (1,UZROS(1),VZROS(1),J1,1=KKK,J)
0092
0093
                          GD TO 140
                     130 WRITE(102,1190) (1,UZROS(1),VZROS(1),J1,UAP((),VAP(1), L=1,JAP)
1f(JAP.LT.J) GO TO 120
140 IF(J.EQ.NG) GO TO 155
0094
0095
0096
                    150 WRITE(102,1095)
0097
```

```
0098
                           IF(J.EQ.0) GO TO 170
0099
                     155 DO 160 [=1,J
UROOT(KO+1)=UZROS(1)
0100
0101
                           VROOT(KD+()=VZROS(()
0102
                           MULTIKD+1)=J1
0103
                           K={J*J1}+K
0104
                           KD=KD+J
0105
                           IF(K-GE-NP) GO TO 1
0106
                     170 J1=J1+1
0107
                           IF(ND1.LE.1) GO TO 200
0108
                           DO 180 [=1.NDI
0109
                           UD0{[]=UD1[[]
0110
                           (1):[0v=(1):00v
                           UDD0([]=UDD1([]
0111
0112
                     180 VDD0(I)=VDD1(I)
0113
                           UDO[ND1+1]=UD1(ND1+1)
0114
                           VD0 (ND1+11=VD1 (ND1+1)
0115
                           ND0=ND1
                           NDD0=NDD1
0116
0117
                           KKK=ND2+1
8110
                           DO 190 I=1.KKK
0119
                           UD1(1)=UD2(1)
0120
                     190 VO1([)=V02([]
                           NO L=ND 2
GO TO 20
0121
0122
0123
                     200 (F(ND1.EQ.0) GO TO 1
0124
                           KD=KD+1
0125
                           DENOM=UD1 (2) *UD1 (21+VO1(2)*VD1(2)
                           URGGT(KD)=(-UD1(1)*UD1(2)-VD1(1)*VD1(2))/DENOM
VRGGT(KD)=(-VD1(1)*UD1(2)+UD1(1)*VD1(2))/DENGM
0126
0127
                           MULTIKO1=J1
0128
                           WRITE(102,3000) (ASTER,1=1,33)
WRITE(102,1035) JL
0129
0130
0131
                           KKK=ND1+1
0132
                           NNN=KKK+1
                           DO 210 I=1,KKK
0133
0134
                           1-NNN=LLL
0135
                     210 WRITE(102,1100) DINAME, ENTRY(JJJ), UDI(JJJ), VD1(JJJ)
0136
                           WRITE(102,1180)
                           WRITE(102,1085) KD. URGOT(KD), VRGGT(KD), J1
0137
                   GO TO 1
1020 FDRMAT(1H1,10X,48HREPEATED USE OF THE GREATEST COMMON DIVISOR AND
0138
0139
                   1020 FDRMAT(IH1,10X,48HREPEATED USE OF THE GREATEST COMMON DIVISOR AND 1,44,44,58H METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIA 2LS/11X,18HPOLYNOMIAL NUMBER ,12//)

1025 FORMAT(//1X,25HNO ROOTS OF MULTIPLICITY ,12//)

1035 FORMAT(///1X,87HTHE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE R 100TS OF P(X) WHICH HAVE MULTIPLICITY ,12//)

1085 FURMAT(2X,5HROOT(,12,4H) = ,023.16,3H + ,D23.16,2H 1,7X,12,18X,25H 1NO [NITIAL APPROXIMATIONS]
0140
0141
0142
0143
                    1095 FORMATI///1X.51HNOT ALL ROOTS OF THE ABOVE POLYNOMIAL.G. WERE FOUN
                          10//)
                   1000 FORMAT(3(12, EX), 9X, 13, 1X, 3(D6.0, 1X), 20X, 2(D7.0, 1X), 11)
1010 FURMAT(2030.0)
0144
0145
                    1015 FORMAT(2030.0)
0146
                   1030 FORMATILX, 22HTHE DEGREE OF PIXI IS , 12, 22H THE COEFFICIENTS ARE//
0147
                         1)
                   1040 FORMAT (2X,A2,A2,4H) = ,D23.16,3H + ,D23.16,2H []
1100 FORMAT (2X,A3,A2,4H) = ,D23.16,3H + ,D23.16,2H []
0148
0149
```

0150	1180 FORMATIV//1x,13HROOTS OF PIX1,52X,14HMULTIPLICITIES,17X,21HINITIAL
0151	1 APPROXIMATION//) 1190 FORMATI2X,5HRJOT(,[2,4H) = ,D23,16,3H + ,D23,16,2H 1,7X,12,7X,D23,
0131	116,3H + ,023,16,2H 1)
0152	2000 FORMATILX,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN. ,12)
0153	2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,[IX,13]
0154	2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,D9.2)
0155	2040 FORMAT(IX, 23HR40TUS TO START SEARCH, 111x, D9, 2)
0156	2050 FORMAT(1x,21HRA01US TO END SEARCH.,13x.D9.2)
0157	2060 FORMAT(//1X)
0158	2070 FORMAT(1x,34HTEST FOR ZERO IN SUBROUTINE GCD. ,D9.2)
0159	2080 FORMAT(1x.34HTEST FOR ZERO IN SUBROUTINE QUAD09.2)
0160	3000 FORMAT(////IX.A3.32A4)
1610	END

```
0001
                        SUBROUTINE PRODUM, UF, VEYN, UG, VG, MN, UH, VHI
                C
                       GIVEN POLYNOMIALS R(X) AND S(X). THIS SUBROUTINE COMPUTES THE COEFFICIENTS OF THE PRODUCT POLYNOMIAL T(X) = R(X).S(X).
               C
               c
0002
                       DOUBLE PRECISION UH, VH, UF, VF, UG, VG
0003
                       DIMENSION UH(51), VH(51), UF(26), VF(26), UG(26), VG(26)
0004
                       MN=M+N
0005
                       KKK=MN+1
0006
                       00 100 I=1.KKK
0007
                       K=!
000B
                       UH( I )=0.0
0009
                       VH( [ ]=0.0
                       IF(I.LE.M+1) GO TO 10
LIMIT=M+1
0010
0011
0012
                       GO TO 20
0013
                   10 LIMIT=1
0014
                   20 00 50 J=1.LIMIT
                       IF(K.GT.N+1) GO TO 50
IF(J+K.EQ.I+1) GO TO 40
0015
0016
0017
                   GO TO 50
40 UH([]=UH([]+(UF(J)+UG(K)-VF(J)+VG(K))
8100
0019
                       VH(I)=VH(I)+(VF(J)+UG(K)+UF(J)*VG(K))
0020
                  100 CONTINUE
0021
0022
                       RETURN
END
0023
```

```
0001
                                SUBROUTINE GENAPPIAPPR, APPI, NAPP, XSTART)
                     00000
                                SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS. WHERE N IS THE DEGREE OF THE ORIGINAL POLYNOMIAL.
                               ODUBLE PRECISION APPR.APPI.XSTART.BETA.
0002
                                                                                                          EPSI.EPSZ.EPS3
                               ODUSE PRECISION APPR, APPI, XS
DIMENSION APPR(25), APPI(25)
COMMON EPSI, EPS2, EPS3, IO2, MAX
IF(XSTART.EQ.D.D) XSTART=0.5
8ETA=0.2617994
DO 10 I=1, NAPP
APPR(1)=XSTART*DCDS(BETA)
0003
0004
0005
0006
0007
0008
0009
                                APPICID=XSTART*DSINIBETAL
                          BETA=BETA+0.5235988
10 XSTART=XSTART+0.5
RETURN
0010
0011
0012
                                EN0
0013
```

```
0001
                                  SUBROUTINE ALTER (XOLDR. XOLDI, NALTER, ITIME)
                      ¢
                      0000
                                 SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT.
                                 DOUBLE PRECISION XOLDR, XOLDI, ABXOLD, BETA, EPS1, EPS2, EPS3
COMMON EPS1, EPS2, EPS3, 102, MAX
IF(111ME.NE.O) GO TO 5
0002
0003
0004
0005
                                  ITIME =1
                             WRITE(102,1010) MAX
5 IF(NALTER.EQ.0) GO TO 10
WRITE(102,1000) XOLDR,XOLDI:
GO TO 20
4000
0007
8000
0009
0010
                            10 ABXOLD=DSQRT((XOLDR+XDLDR)+(XOLD1+XOLD1))
0011
                                 BETA=DATAN2(XOLD [ . XOLDR ]
0012
                                 WRITE(102,1020) XOLOR, XOLDI
                           20 NALTER=NALTER+1
IF(NALTER-GT-5) RETURN
GO TO (30,40,30,40,30).NALTER
30 XULDR=-XOLDR
0013
0014
0015
0016
0017
                                 XOLDI=-XOLDI
                                GU TO 50
BETA=BETA+1.0471976
XOLOR=ABXOLD+DCOS(BETA)
XOLDI=ABXOLD+OSIN(BETA)
0018
0019
0020
0021
0022
                            50 RETURN
                        1000 FORMAT(1x,023.16,3H + ,023.16,2H I,10x,21HALTERED APPROXIMATION)
1010 FORMAT(///1x,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
LTER ,13,12H ITERATIONS.//)
1020 FORMAT(/1x,023.16,3H + ,023.16,2H I,10x,21HINITIAL APPROXIMATION)
0023
0024
0025
0026
                                 END
```

```
1000
                      SUBROUTINE ZERDS(NQ, UQ, VQ, NAPP, UAPP, VAPP, J, UROOT, VROOT, JAP, UAP, VAP
                     1.ENTRY.XSTART, XEND
                     NEWTONS METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A
              000000
                     POLYNOMIAL OF MAXIMUM DEGREE 25 BY COMPUTING A SEQUENCE OF APPROX-IMATIONS CONVERGING TO A ZERO OF THE POLYNOMIAL USING THE ITERATION
                     FORMULA
                                     X(N+1) = X(N)-P(X(N))/P^*(X(N)).
0002
                     DOUBLE PRECISION UAPP. VAPP. URGOT, VROOT, ÚZRO, VZRO, UQ. VQ. UDUMMY, VDUM
                     1MY, UQQ, VQQ, UAP, VAP, UQD, VQD, URQOTS, VROOTS, EPS1, EPS2, EPS3, UAPROX, VAP
                    2ROX
0003
                     DOUBLE PRECISION XEND, XSTART
DIMENSION UAPP(251, VAPP(25), UROOT(251, VROOT(25), UQ(26), VQ(26), UQQ(
0004
                     1261, VQQ (261, UAP (251, VAP (25), UQO (26), VQO (26), ENTRY (26), URQQT $(25), V
                    2RGOT5(25)
0005
                     COMMON EPS1, EPS2, EPS3, TO2, MAX
                     DATA QQNAME, QNAME/3HQQI, 2HQI/
LOGICAL CONV
0006
0007
0008
                      O=L
0609
                      ITIME=0
0010
                      1F(NQ.GE.3) GO TO 85
0011
                     GO TO 110
0012
                  B5 KKK=NQ+1
0013
                     00 90 I=1,KKK
0014
                     ugg(I)=ug(I)
0015
                     VQQ{[]=VQ(])
0016
                     NOQ=NQ
0017
                     GO TO 120
                110 CALL QUAD(NQ,UQ,VQ,J,URODT,VROOT)
0018
0019
                     JAP=0
0020
                     GO TO 310
0021
                 120 00 200 [=1,NAPP
0022
                     IALTER=0
                     HAPROX=UAPPLID
0023
                     VAPROX=VAPP(I)
0024
0025
                130 CALL NEWTON (UAPROX, VAPROX, NQQ, UQQ, VQQ, UZRO, VZRO, CONV)
0026
                     IFICONV) GO TO 160
0027
                     CALL ALTER(UAPP(I), VAPP(I), IALTER, ITIME)
0028
                     IF(IALTER.GT.5) GO TO 200
                     (1)99AH=XOR9AH
0029
                     VAPROX=VAPP(I)
0030
                     60 10 130
0031
0032
                160
                     J=J+1
                     UROOT (J) = UZRO
0033
0034
                     VROUT(J)=VZRO
                     UAP(J)=UAPROX
0035
                     VAP(J) = VAPROX
0036
                     CALL HORNER LUZRO, VZRO, NQQ, UQQ, VQQ, UQD, VQD, UDUMMY, VDUMMY)
0037
                     00 180 I1=1,NQQ
UQQ([1]=UQD([1+1]
0038
0039
                180 VQQ([1]=VQD([1+1]
0040
0041
                     NCO=NOO-1
                     IF(NQQ.GE.3) GO TO 200
0042
0043
                     GO TO 220
0044
```

```
0045
                   200 CONTINUE
                        IF(J.GE.NQ) GO TO 205
IF(XEND.EQ.O.O) GO TO 205
IF(XSTART.GT.XEND) GO TO 205
0046
0047
0048
0049
0050
                        CALL GENAPP(UAPP, VAPP, NAPP, XSTART)
                   GO TO 120
205 [F(N90-LE-2) GO TO 210
WRJTE(102,1200)
0051
0052
0053
0054
                        KKK=NOQ+1
0055
                        NNN=KKK+ L
                   DG 157 L=1,KKK

JJJ=NNN-L

157 WRITE(102,1100) QQNAME,ENTRY(JJJ),UQQ(JJJ),VQQ(JJJ)
0056
0057
0058
                   210 IF(J.EQ.0) GO TO 310
0059
0060
                        L=9AL
0061
                        GO TO 230
                  220 CALL QUAD(NQQ,UQQ,VQQ,J,URQQT,VRQQT)
230 WRITE(102,1132)
WRITE(102,1133) (1,URQQT(1),VRQQT(1),UAP(1),VAP(1),I=L,JAP)
1F(JAP-LT-J) GO TO 235
0062
0063
0064
0065
0066
                        GO TO 240
0067
                   235 KKK=JAP+1
                        WRITE(102,1134) (I,UROOT([],VROOT([],[=KKK,J]
0068
0069
                   240 J1=0
                        DO 300 [=1.J
0070
0071
                        CALL NEWTON(UROOT(1), VROOT(1), NQ, UQ, VQ, UZRO, VZRO, CONV)
                        IF(CONV) GO TO 280
WRITE(102,1140) I,UROOT(1),VROOT(1),MAX,NQ
0072
0073
0074
                        KKK=NQ+1
                        NNN=KKK+1
0075
0076
                        DO 242 L=1,KKK
0077
                         J-MMM-L
                   242 HRITE(102,1040) QNAME,ENTRY(JJJ),UQ(JJJ),VQ(JJJ)

IF(I.LT.JAP) GO TO 241

IF(I.EQ.JAP) GO TO 250

GO TO 300
0078
0079
0080
0081
0082
                   241 KKK#JAP-1
                        DO 245 [1=1.KKK
UAP([1]=UAP([1+1]
0083
0084
                  245 VAPITII=VAPITI+11
250 JAP=JAP-1
0085
0086
0087
                        GO TO 300
                        J1=J1+1
0088
0089
                        URBOTS(J11=U2RO
0090
                        VROOTS (JI I=VZRO
                   300 CONTINUE
0091
                        j=Jl
0092
0093
                        IF(J.EQ.0) GO TO 305
                        00 303 [=1.J
0094
                        URBOT([])=URBOTS([])
0095
                  303 VRUOT(1)=VROOTS(1)
0096
                   GO TO 310
305 WRITE(102,1150) NQ
0097
0098
0099
                        KKK=NQ+1
0100
                        NNN=KKK+1
0101
                        00 306 L=1.KKK
                        JJJ=NNN-L
0102
```

	·
0103	(LLLIGY, (LLL) YRTH3.3MAND (0.01.101) 4000
0104	310 RETURN
0105	1200 FORMATI///1x,7OHCOEFFICIENTS OF THE DEFLATED POLYNOMIAL FOR WHICH 1NO ZEROS WERE FOUND.//)
0106	1132 FORMATI///1x,13HROOTS OF G(X),84X,21HINITIAL APPROXIMATION//)
0107	1133 FORMAT(2X,5HROOT(,[2,4H) = ,023.16,3H + ,023.16,2H [,17X,D23.16,3H 1 + ,023.16,2H ])
0108	1134 FORMAT(2X,5HROOT(,12,4H) = ,023,16,3H + ,023,16,2H 1,22X,26HRESULT 1S OF SUBROUTINE QUAD)
0109	1140 FORMAT(///,1x,40HND ROOTS FOR INITIAL APPROXIMATION ROOT(,12,4H) = 1 .D23.16,3H + .D23.16,2H I/6H AND .T3,40H ITERATIONS ON THE POLYN 2DMIAL OF DEGREE .I2.18H WITH COEFFICIENTS//)
0110	1150 FORMAT(//, 1x, 45HNO ROOTS FOR THE POLYNOMIAL Q(X) OF DEGREE = ,12, 138H WITH GENERATED INITIAL APPROX(MATIONS//)
0111	1040 FORMAT (2X,A2,A2,4H) = .D23.16,3H + .D23.16,2H 1)
0112	1100 FORMAT(2x,43,42,4H) = .023.16.3H + .D23.16.2H ()
0113	END

```
0001
                     SUBROUTINE GCDIN.UR.VR.M.US.VS.M1.USS.VSSI
                   * GIVEN POLYNOMIALS P(X) AND DP(X) WHERE DEG. DP(X) IS LESS THAN DEG. * P(X). SUBROUTINE GCD COMPUTES THE GREATEST COMMON DIVISOR OF PIX) AND
              Ċ
              000
                     DP(X).
0002
                     DOUBLE PRECISION USSSSS. VSSSSS
0003
                     DOUBLE PRECISION UR, VR, US, VS, USS, VSS, URR, VRR, UD, VD, UT, VT, EPSLON, EP
                    152,EP53,BBB
DIMENSION UR(26),VR(26),US(26),VS(26),USS(26),VSS(26),URR(26),VRR(
0004
                    126), UT (26), VT(26)
0005
                     COMMON EPSLON, EPS2, EPS3, TOZ, MAX
0006
                     N1=N
0007
                     M1 * M
                     KKK=N+1
OOOR
0009
                     DO 20 1=1.KKK
                     URR([]=UR([]
0010
0011
                     VRR(I)=VR(I)
0012
                     KKK=M+1
                     DO 25 I=1.KKK
USS(I)=US(I)
0013
0014
                     VSS(1)=VS(1)
0015
                 30 BBB=USS(M1+1)*USS(M1+1)+VSS(M1+1)*VSS(M1+1)
0016
0017
                     UD=(URR(N1+11+USS(M1+11+VRR(N1+11+VSS(M1+111/888
0018
                     VD=(USS(M1+1)*VRR(N1+11-URR(N1+1)*VSS(M1+1))/888
0019
                     KKK=N1+1-M1
                     DO 40 |=KKK,NL
UT(|)=URR(|)-(UD#USS(|-N|+M1)-VD#VSS(|-N|+M1))
VT(|)=VRR(|)-(UD#VSS(|-N|+M1)+VD#USS(|-N|+M1))
0020
0021
0022
0023
                     IF(M1.EQ.N1) GO TO 70
0024
                     KKK=N1-M1
0025
                     DO 60 [=],KKK
UT(I)=URR(I)
0026
                 60 VT(1)=VRR([]
0027
0028
                  70 DO 90 [=1,N]
0029
                     888=02QRY(UT(N1+1-1)*UT(N1+1-11+YT(N1+1-1)*VT(N1+1-11)
0030
                     IF(8BB.GT.EPSLON) GO TO 100
                 90 CONTINUE
0031
                     DO 95 1=1.81
0032
0033
                     BBB=USS(M1+11*USS(M1+1)+VSS(M1+1)*VSS(M1+1)
0034
                     USSSSS=(USS(1)*USS(M1+1)+VSS(1)*VSS(M1+1))/BBB
0035
                     888\(11+1M)22V*(1)2ZV*(1+1M)2ZV+1=2222ZV
0036
                     US5(1)=USSSSS
                 22222V=(1)22V 26
0037
0038
                     USS(ML+1)=1.0
0039
                     VSS(M1+1)=0.0
0040
                     GO TO 200
0041
                100 K=N1-1
                     IF(K-EQ.0) GO TO, 170
0042
                     IFIK-LY-MLF GO TO 140
0043
0044
                     KKK=K+1
0045
                     DO 130 J=1,KKK
0046
                     {L}TU=(L]¶NU
                L30 VRR(J)=VT(J)
0047
                     N1≠K
0048
```

GG TO 30

0049

140	KKK=K+1
	DO 150 J=1.KKK
	URR(J)=USS(J)
	(L)22V=(L)ARV
	USS(J)=UT(J)
150	(L)TV=(L)22V
	KKK=K+2
	NNN=M1+1
	DO 160 J=KKK,NNN
	URR(J1=USS(J1
160	{L}22V={L}33V
	NI=MI
	M1=K
	GD TO 30
170	USS(1)=1.0
	VSS(1)=0.0
	M1=0
200	RETURN
•	END
	150 160 170

```
0001
                       SUBROUTINE NEWTON (UX, VX, N, UP, VP, UXO, VXO, CONV)
               C
               000000
                       THIS SUBROUTINE CALCULATES A NEW APPROXIMATION FROM THE OLD APPROX-
                      IMATION BY USING THE ITERATION FORMULA

X(N+1) = X(N)-P(X(N))/P*(X(N)).
0002
                       DOUBLE PRECISION UX, VX, UP, VP, UXD, VXO, U8, V8, UDPXO, VDPXO, UPXO, VPXO, U
                      1D1FF, VOIFF, EPS1, EPSLON, EPS3, AAA, BBB
                       DOUBLE PRECISION DDD
DOUBLE PRECISION ABPXO
DIMENSION UP(26), VP(26), UB(26), VB(26)
0003
0004
0005
0006
                       COMMON EPS1. EPSLON, EPS3. 102. MAX
0007
                       LOGICAL CONV
8000
                       XU≖OXU
0009
                       AX0=AX
                       DO 10 I=1.MAX
CALL HORNER(UXO,VXO,N,UP,VP,UB,VB,UDPXO,VDPXO)
0010
0011
0012
                       UPXD=UB(1)
0013
                       VPXO=V8(L)
0014
                       DDD=DSQRT(UDPXQ+UDPXQ+VDPXQ+VDPXQ)
                       IF(DDD.NE.O.O) GO TO 5
ABPX0=DSQRT(UPXO+UPXO+VPXO+VPXO)
0015
0016
                       IF(A8PXO.EQ.O.O) GO TO 20
0017
0018
                       GO TO 15
0019
                    5 888=UDPX 0*U0PX 0*VDPX 0*VDPX 0
0020
                       UD1FF=(UPXG*UDPXG+VPXG*VDPXG)/888
                       VDIFF=(VPXO*UDPXO-UPXO*VDPXO)/888
0021
                       UX0=UX0-U01FF
VX0=VX0-V01FF
0022
0024
                       AAA=OSQRT(UD]FF*UDIFF+VD[FF*VD[FF}
0025
                       BRB=DSQRT(UXO*UXO+VXO*VXO)
IF(BBB.EQ.O.O) GO TO 10
IF(AAA/888.LT.EPSLON) GO TO 20
0026
0027
0028
                   10 CONTINUE
0029
                   15 CONV=.FALSE.
0030
                       RETURN
0031
                   20 CONV=.TRUE.
0032
                       RETURN
```

END.

0033

```
0001
                               SUBROUTINE DIVIDEIN, UP, VP, M, UD, VD, K, UQ, VQ)
                    000000
                              GIVEN TWO POLYNOMIALS F(x) AND G(x), SUBROUTINE DIVIDE COMPUTES THE QUOTIENT POLYNOMIAL H(x) = F(x)/G(x).
                              DOUBLE PRECISION UP, VP, UD, VD, UQ, VQ, UTERM, VTERM, UDUMMY DIMENSION UP(26), VP(26), UD(26), VD(26), UQ(26), VQ(26)
0002
0003
0004
0005
                              UDUMMY=UD(M+1)*UD(M+1)+VD(M+1)*VD(M+1)
                              AG(K+1)=(Ab(U+1)+AD(W+1)+Ab(W+1)+AD(W+1))ADDMWA
AD(K+1)=(Ab(U+1)+AD(W+1)+Ab(W+1)+AD(W+1))ADDMWA
0006
0007
                              IF(K.EQ.0) GO TO 100
0008
0009
                              .1 = -1
0010
                              DO 50 1=1,K
                              J=J+1
UTERM=UP(N-J)
VTERM=VP(N-J)
0011
0012
0013
                              KK=K+1
0014
0015
                              NNN=M-J
0016
                              DO 40 M1=NNN.H
0017
                               IF(KK.GT.11 GO TO 10
0018
                              GO TO 45
                         10 IF(M1.GE.1) GO TO 20
GO TO 40
20 UTERM-UTERM-(UQ(KK)*UD(M1)-VQ(KK)*VD(M1)1
VTERM=VTERM-(UQ(KK)*VD(M1)+VQ(KK)*UD(M1))
0019
0020
0021
0022
0023
                         45 UDUMMY=UD(M+1)*UD(M+1)*VD(M+1)*VD(M+1)
UQ(K+1-1)=(UTERM*UD(M+1)*VTERM*VD(M+1))/UDUMMY
50 VQ(K+1-1)=(VTERM*UD(M+1)-UTERM*VD(M+1))/UDUMMY
0024
0025
0026
                       100 RETURN
0028
                              END
```

```
0001
                                              SUBROUTINE HORNERIUX, VX, N, UP, VP, UB, VB, UC, VC)
                              0000000
                                         * HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A * POINT D AND ITS DERIVATIVE AT D. SYNTHETIC DIVISION IS USED TO * DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE FACTOR (X - D).
                                             DOUBLE PRECISION UX,VX,UP,VP,UB,VB,UC,VC
DOUBLE PRECISION UDUMMY,VDUMMY
OIMENSION UP(26),VP(26),UB(26),VB(26)
0002
 0003
0004
                                             Olympia (nx+ng (n+1)+Ax*ng (n+1))+Ab(u)

ng(u+1)=Ab(u+1)

ng(u+1)=Ab(u+1)

ng(u+1)=Ab(u+1)+Ax*ng(u+1))+Ab(u)

ng(u+1)=Ab(u+1)+Ax*ng(u+1))+Ab(u)
0005
0006
0007
8000
0009
                                              UC=UB(N+1)
0010
                                              VC=V8[N+1]
                                            VC=VDUMMY+V8!KKK+2-1)
PFTURN

VKK+1-10
VB(KKK+1-1)=(UX*U8(KKK+2-1)-YX*VB(KKK+2-1))+VP(KKK+1-1)
VB(KKK+1-1)=(UX*VB!KKK+2-1)+VX*UB(KKK+2-1))+VP(KKK+1-1)
VDUMMY=UX*VC-YX*VC
VDUMMY=UX*VC-YX*VC
VDUMMY+VB!KKK+2-1)
VC=VDUMMY+VB!KKK+2-1)
PFTURN
0012
0013
0014
0015
0016
0017
0018
0019
0020
                                              END
```

```
0001
                          SUBROUTINE QUADIN-UA-VA-J-URGOT-VROOTI
                 CCC
                          SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR. SOLUTION OF THE QUADRATIC IS DONE USING THE QUADRATIC FORMULA.
                         DOUBLE PRECISION EPS1, EPS2, EPSL ON, UROOT, VROOT, UA, VA, UDISC, VDISC, UD
0002
                        1,VD,ODD,UTEMP,VTEMP,BBB
OIMENSION URODT(25),VRODT(25),UA(26),VA(26)
0003
0004
                          COMMON EPS1, EPS2, EPSLON, TO2, MAX
0005
                          IF(N.GT.1) GO TO 10
0006
                         J=J+1
BB=UA(2)*UA(2)*VA(2)*VA(2)
0007
8000
                         UROGT(J) =- (UA(L) +UA(2)+VA(L) +VA(2)1/B88
0009
                          VRGCT(J)=-(VA(1)+UA(2)-UA(1)+VA(2))/888
                     GO TO 100

10 UDISC=(UA(2)*UA(2)-VA(2)*VA(2))-(4.0*(UA(3)*UA(1)-VA(3)*VA(1)))

VDISC=(2.0*UA(2)*VA(2))-(4.0*(UA(3)*VA(1)+VA(3)*UA(1)))

UD=2.0*UA(3)
0010
0011
0012
0013
                         VD=2.0*VA(3)
0014
0015
                         DDD=DSQRT(UDISC*UDISC*VDISC*VDISC)
0016
                          IF(ODD.LT.EPSLON) GO TO 20
                         CALL COMSQT(UDISC, VOISC, UTEMP, VTEMP)
0017
0018
                         B88=U0*UD+V0*VD
0019
                         URGOT(J+1)=((-UA(2)+UTEMP)+UD+(-VA(2)+VTEMP)+VD)/BBB
                         VROOT(J+1)=[(-VA(2)+VTEMP)*UD-(-VA(2)+UTEMP)*VD)/BBB
URGOT(J+2)=((-VA(2)-UTEMP)*UD+(-VA(2)-VTEMP)*VD)/BBB
0020
0021
0022
                         VROOT(J+2)=((-VA(2)-VTEMP)#UD-(-UA(2)-UTEMP)#VO)/888
0023
                         J=J+2
GO TO LOO
0024
0025
0026
                         BBB=UD*UD+VD*VD
                         URODT(J)=(-UA(2)*UD-VA(2)*V0)/888
VROOT(J)=(-VA(2)*UD+UA(2)*VD)/888
0027
0028
                  WRITE([02,1000) UROUT(J), VROOT(J)
1000 FORMAT(///LX,11HQUAD FOUND ,D23.16,3H + .D23.16,2H 1,22H TO BE A M
0029
0030
                       IULTIPLE ROOT//)
0031
                    100 RETURN
0032
                         END
```



```
1000
                         SUBROUTINÉ DERIVINAUP, VP, M, UA, VA)
                000000
                        GIVEN A POLYNOMIAL PIXI, SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF
                      . ITS DERIVATIVE P'(X).
                        DOUBLE PRECISION UP. VP. UA. VA. AAA
DIMENSION UP(26), VP(26), VA(26), VA(26)
0002
0003
0004
                    00 to 1=2,KKK
AAA=[-1
UA(1-1)=AAA+UP(1)
10 VA(1-1)=AAA+VP(1)
0005
0006
0008
0009
                        M=N-1
0010
                        RETURN
0011
                        END
```

```
0001
                           SUBROUTINE COMSQTIUX, VX, UY, VY)
                 00000
                          THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.
                          DOUBLE PRECISION UX.VX,UY.VY,DUMMY,R.AAA.888
R=DSQRT(UX*UX+VX*VX)
AAA=DSQRT(DABS((R+UX)/2.01)
BBB=DSQRT(DABS((R-UX)/2.01)
0002
0003
0004
0005
0006
                           IF(VX) 10,20,30
0007
                      10 UY#AAA
                      VY=-1.0*888
GO TO 100
20 1F(UX) 40,50,60
8000
0009
0010
0011
                      30 UY=AAA
0012
                           VY = 6.88
                      GO TO 100
40 DUMMY=DABS(UX)
UY=0.0
0013
0014
0015
0016
                           VY=DSQRT(DUMMY)
0017
                          GO TO 100
001B
                      50 UY=0.0
                      VY=0.0

GU TO 100

60 DUMMY=DABS(UX)
0019
0020
0021
0022
                          UY=DSQRT(DUMMY)
0023
                           VY=0.0
Ö024
                          RETURN
0025
                          END
```

#### APPENDIX H

#### REPEATED G.C.D. - MULLER'S METHOD

#### 1. Use of the Program

A double precision FORTRAN IV program using the repeated G.C.D. method with Muller's method as a supporting method is presented here. Flow charts for this program are given in Figure H.1 while Table H.III gives a FORTRAN IV listing of this program.

This program is designed to solve polynomials having degree less than or equal to 25. In order to solve polynomials of degree N where N > 25, the data statement and array dimensions given in Table H.I must be changed.

In this program both the leading coefficient and the constant coefficient are assumed to be non-zero.

#### TABLE H.I

PROGRAM CHANGES NECESSARY TO SOLVE POLYNOMIALS OF DEGREE GREATER THAN 25 BY THE REPEATED G.C.D. - MULLER'S METHOD

#### Main Program

Data Entry/lH1,1H2,...,1H9,2H10,2H11,...,2HXX/where XX = N+1UAPP(N,3), VAPP(N,3)URAPP(N,3), URAPP(N,3)UP(N+1), VP(N+1)MULT(N) UDDO(N+1), VDDO(N+1)UD1(N+1), VD1(N+1)UDD1(N+1), VDD1(N+1)UD2(N+1), VD2(N+1)UG(N+1), VG(N+1)UD3(2N+1), VD3(2N+1)UD4(2N+1), VD4(2N+1)UAP(N+1), VAP(N+1)UZROS(N), VZROS(N)
UROOT(N), VROOT(N) UDO(N+1), VDO(N+1) ENTRY (N+1)

Subroutines PROD, QUAD

See corresponding subroutine in Table G.I.

Subroutines DERIV, GCD, and DIVIDE

See corresponding subroutine in Table E.I.

Subroutines MULLER, GENAPP, BETTER and HORNER

See corresponding subroutine in Table F.I.

2. Input Data for Repeated G.C.D. - Muller's Method

The input data to the repeated G.C.D. - Muller's method is the same as for the repeated G.C.D. - Newton's method as described in Appendix G, § 2.

3. Variables Used in Repeated G.C.D. - Muller's Method

The variables used in this program are referenced in Table H.II.

The notation and symbols used in the referenced tables are described in Appendix E, § 3.

#### TABLE H.II

VARIABLES USED IN REPEATED G.C.D. - MULLER'S METHOD

Main Program and Subroutine PROD

See Table G.II.

Subroutines QUAD, DERIV, GCD, DIVIDE, and COMSQT See corresponding subroutine in Table E.VI.

Subroutines CALC, MULLER, GENAPP, ALTER, BETTER, TEST, and HORNER

See corresponding subroutine in Table F.II.

#### 4. Description of Program Output

The output for this program is the same as that for repeated G.C.D. - Newton's method as described in Appendix G, § 4. Only one initial approximation,  $X_0$ , (not three) is printed. The other two required by Muller's method are  $.9X_0$  and  $1.1X_0$ . The message "SOLVED BY DIRECT METHOD" means that the corresponding root was obtained by Subroutine QUAD.

# 5. Informative Messages and Error Messages

Descriptions of the informative messages and error messages printed by this program can be found either in Appendix E, § 5, Appendix F, § 5, or Appendix G, § 5.

# MAIN PROGRAM

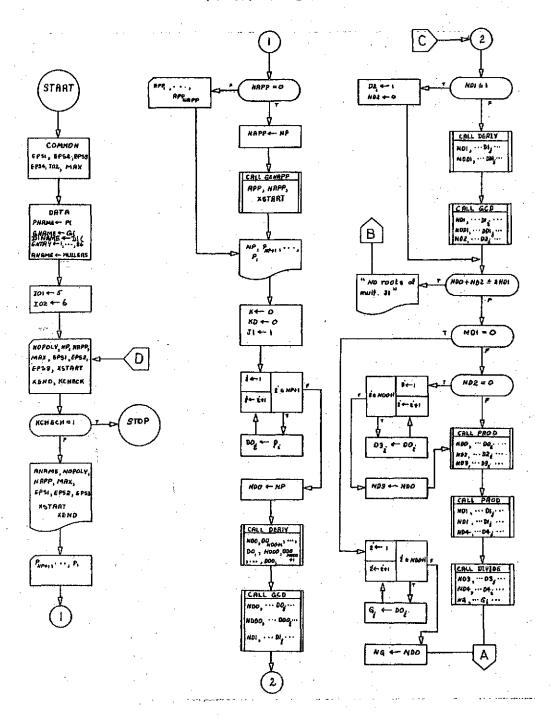


Figure H.1. Flow Charts for Repeated G.C.D.-Muller's Method

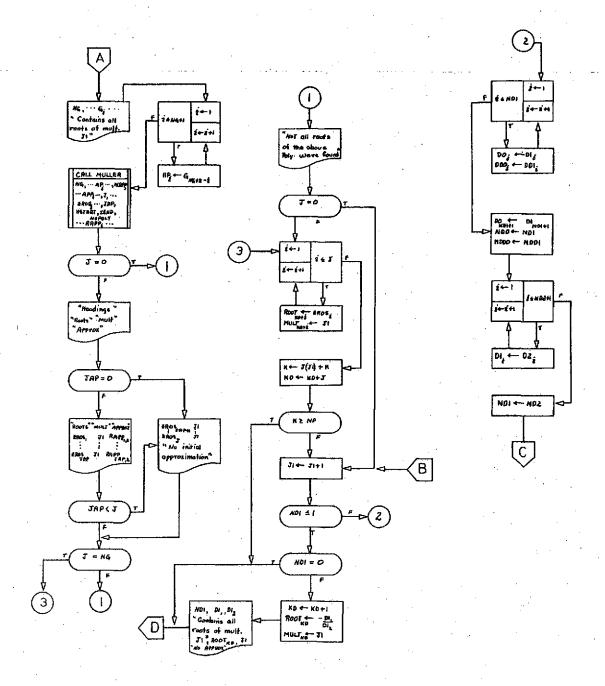


Figure H.1. (Continued)

# MULLER

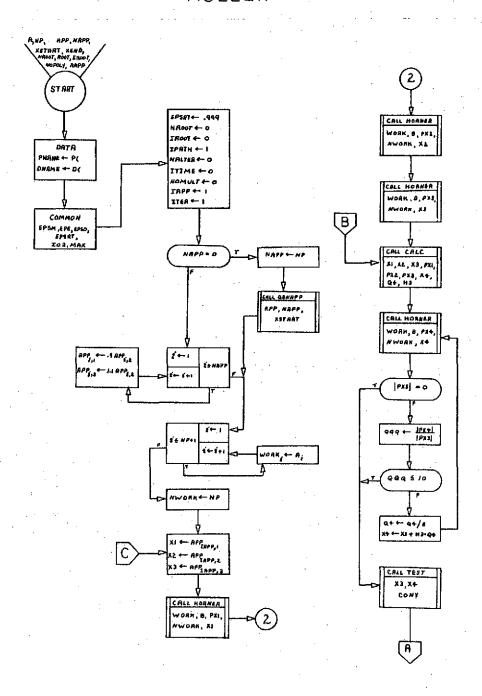


Figure H.1. (Continued)

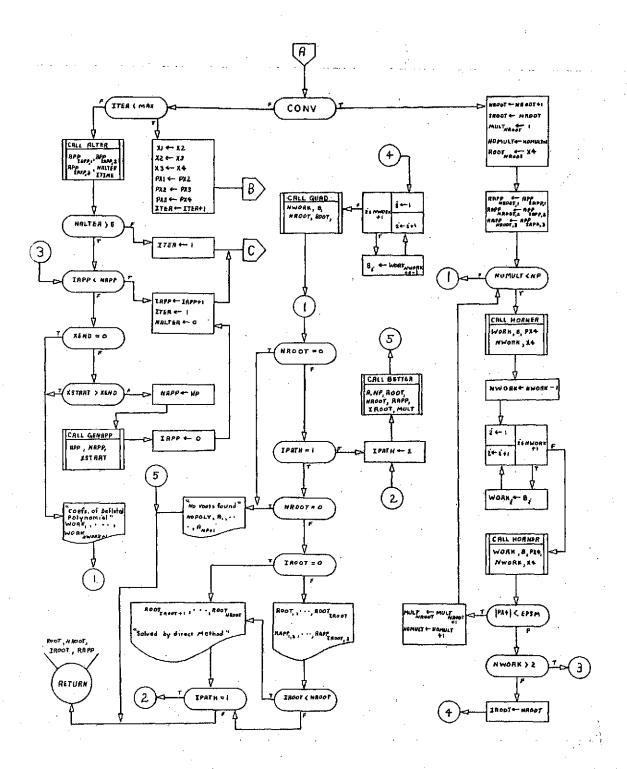
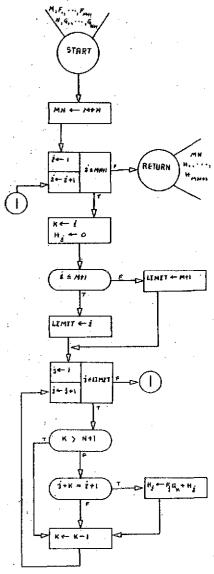


Figure H.1. (Continued)

PROD

QUA D



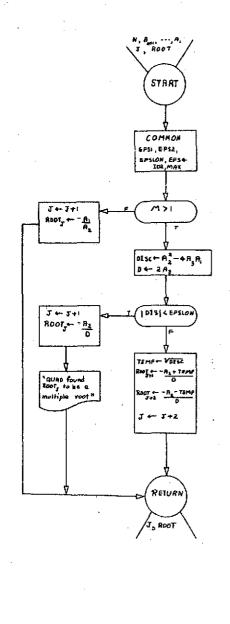
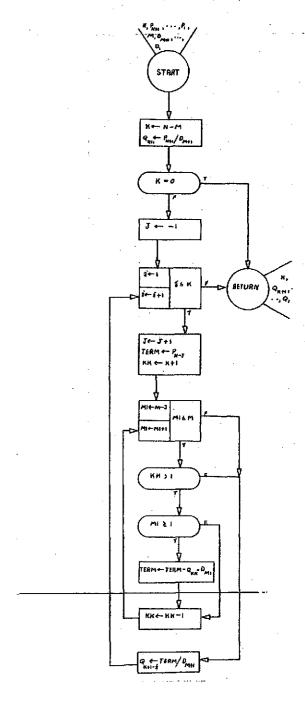


Figure H.1. (Continued)

# DIVIDE



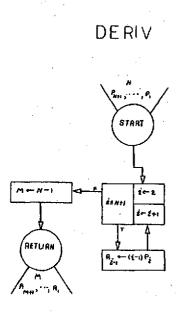


Figure H.1. (Continued)

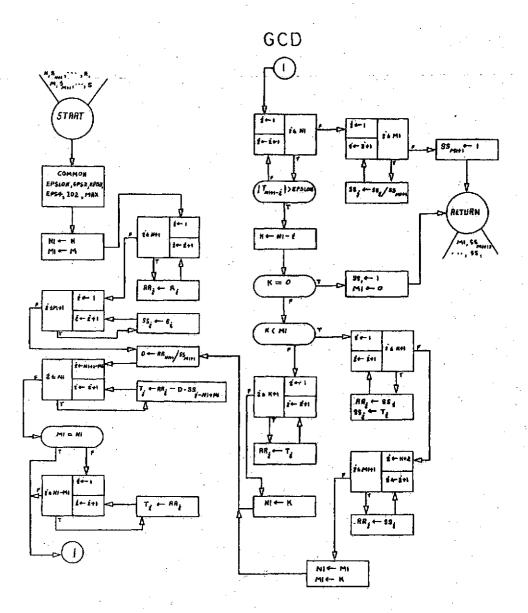
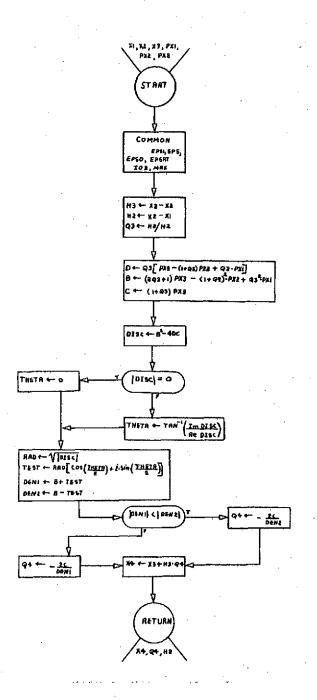


Figure H.1. (Continued)



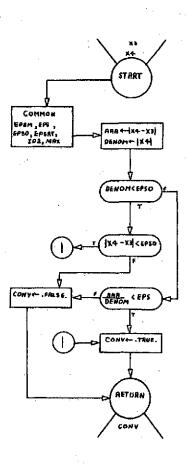


Figure H.1. (Continued)

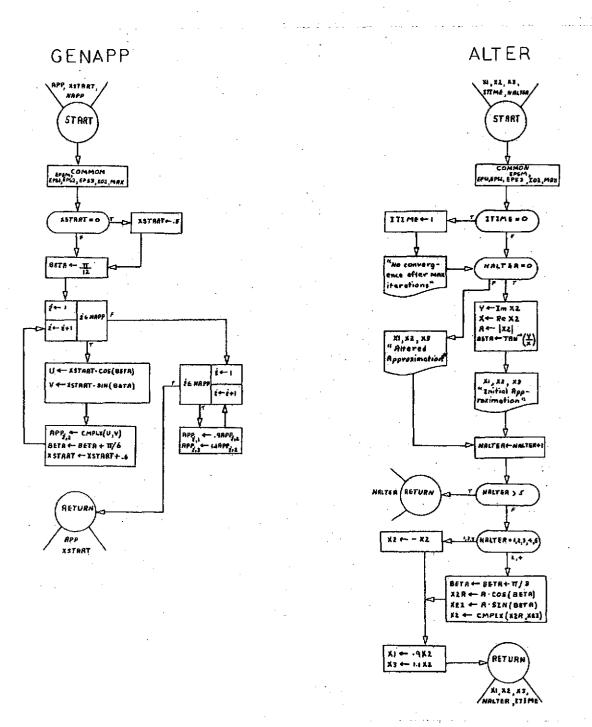


Figure H.1. (Continued)

- mary and the

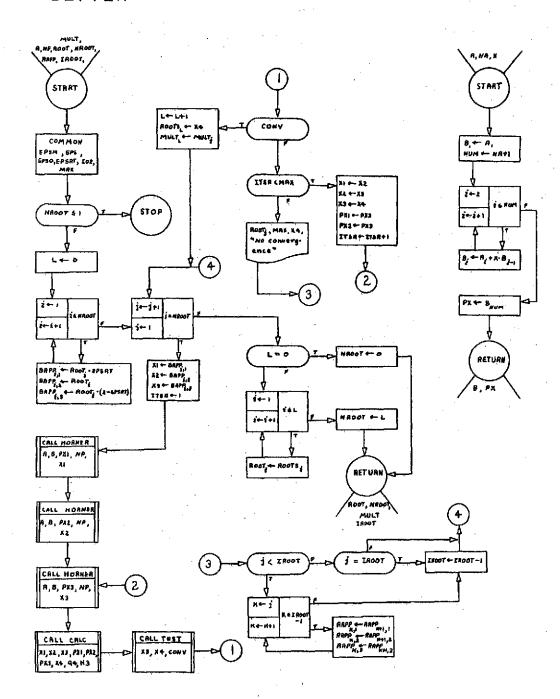
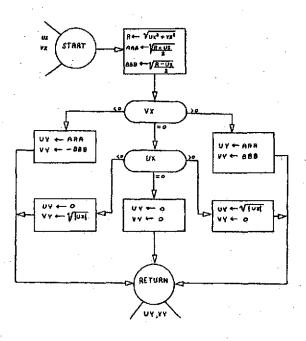


Figure H.1. (Continued)

# COMSQT



Figue H.1. (Continued)

#### TABLE H.III

### PROGRAM FOR REPEATED G.C.D.-MULLER'S METHOD

```
DOUBLE PRECISION PROGRAM FOR THE REPEATED G.C.D. - MULLER'S METHOD
                    0000
                           * THIS METHOD REPEATEDLY FINDS THE GREATEST COMMON DIVISOR OF TWO * POLYNOMIALS IN ORDER TO EXTRACT THE ZEROS IN GROUPS ACCORDING TO * MULTIPLICITY USING NEWTON'S METHOD. ALL ZEROS OF MULTIPLICITY 1 * ARE EXTRACTED FOLLOWED BY THOSE OF MULTIPLICITY 2, ETC.
                             OGUBLE PRECISION EPSI, EPS2, EPS3, UP, VP, UAPP, VAPP, UDO, VDO, UDDO, VDDO, LUO1, VOI, UD2, VOZ, UDD1, VDD1, UG, VG, UD3, VO3, UD4, VD4, UZRUS, VZROS, UAP, VA 2P, URODT, VROOT, DENOM DOUBLE PRECISION XSTART DOUBLE PRECISION XEND
0001
0002
0003
0004
                              DOUBLE PRECISION URAPP, VRAPP
0005
                              DOUBLE PRECISION EPS4
                             DIMENSION UAPP(25,3), VAPP(25,3), URAPP(25,3), VRAPP(25,3)
DIMENSION UP(26), VP(26), MULT(25), UDDO(26), VDOO(26), UD1(26), VO1(26), UD2(26), VD2(26), UG(26), VG(26), UG3(51), VD3(51), U
204(51), VD4(51), UAP(26), VAP(26), UZROS(25), VZROS(25), URDOT(25), VROOT
0006
0007
                             3(25), ANAME(2), UDO(26), VDO(26), ENTRY(26)
COMMON EPS1, EPS2, EPS3, EPS4, IO2, MAX
DATA PNAME, GNAME/2HP1, 2HG(/, DLNAME/3HD1(/
8000
0009
                              DATA ASTER/4H***/
0010
                             DATA ENTRY/INI.1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,2H10,2H11,2H12,2H13
1,2H14,2H15,2H16,2H17,2H18,2H19,2H20,2H21,2H22,2H23,2H24,2H25,2H26/
0011
0012
                              DATA ANAME(1), ANAME(2)/4HMULL, 4HERS /
0013
                              101=5
0014
                              102 * 6
                           1 READ(101,1000) NOPOLY, NP, NAPP, MAX, EPS1, EPS2, EPS3, XSTART, XEND, KCHEC
0015
                             ìΚ
0016
                              IF(KCHECK.EQ.1) STOP
0017
                              WRITE(102,1020) ANAME(1), ANAME(2), NOPOLY
0018
                              WRITE(IDZ,2000) NAPP
                              WRITE(102,2010) MAX WRITE(102,2070) EPSL
0019
0020
                              WRITE! 102,20201 EPS2
0021
                              WRITE(102,2080) EPS3
0022
0023
                              WRITE(ID2,2040) XSTART
                              WRITE(102,2050) XEND
0024
0025
                              WRITE(102,2060)
                              KKK=NP+1
0026
0027
                              NNN≖KKK+1
                              DO 5 I=1,KKK
0028
0029
                           5 READ(101,1010) UP(JJJ), VP(JJJ)
1F(NAPP.NE.O) GO TO 22
0030
0031
0032
                              CALL GENAPPIUAPP, VAPP, NAPP, XSTARTI
0033
                         GO TO 23
22 READ([O1,1015] (UAPP([,2],VAPP([,2],[=1,NAPP)
0034
0035
0036
                         23 WRITE(102,1030) NP
                              KKK=NP+1
0037
                              NNN=KKK+1
0038
                              DO 8 I=1.KKK
0039
                              1-444=
```

```
-0041
                      8 WRITE(102,1040) PNAME, ENTRY(JJJ), UP(JJJ), VP(JJJ)
0042
                        K=0
0043
                        KD=0
0044
                         J1=4
                        KKK=NP+1
0045
0046
                        DO 10 1=1.KKK
0047
                         U001[]=UP([]
0048
                    10 VD01[]=VP[]]
0049
                        ND0=NP
                    CALL DERIV(NDO,UDO,VDO,NDOO,UDDO,VDDO)
CALL GCD(NDO,UDO,VDO,NDDO,UDDO,VDOO,NDE,UD1,VDI)
20 WRITE(102,3000) (ASTER,1=1,33)
IF(NDI,LE-1) GO TO 30
0050
0051
0052
0053
0054
                    GO TO 40
30 UD2(11=1.0
0055
                        VD2(1)=0.0
0056
0057
                        ND2=0
0058
                        GO TO 50
0059
                    40 CALL DERIVINDI, UDI, VDI, NDDI, UDDI, VDDI)
                       CALL GCD(NO1, UD1, VD1, NO01, UDD1, VD01, ND2, UD2, VD2)
IF(ND0+ND2, LE, 2*ND1) GD TD 60
GD TD 70
WRITE(102, 1025) J1
0060
0061
0062
0063
0064
                        GB TO 170
0065
                        IF(ND1.EQ.0) GO TO 80
                    GO TO 90
80 KKK=N00+1
0066
0067
0068
                        DO 85 [#1,KKK
0069
                        UG(1)=UDO(1)
0070
                        VG([]=VDO([]
0071
                        NG=NDO
0072
                    GO TO 110
90 [F(ND2.EQ.0) GO TO 115
0074
                        CALL PROD (NDO, UDO, VDO, NDZ, UDZ, VD2, ND3, UD3, VD3)
0075
                   100 CALL PROD(NO1, UD1, VD1, ND1, UD1, VD1, ND4, UD4, VD4)
0076
                        CALL DIVIDE(ND3, UD3, VD3, NO4, UD4, VD4, NG, UG, VG)
                  110 WRITE(102,1035) J1
0077
                        KKK=NG+1
0078
0079
                        NNN=KKK+L
0080
                        DO 112 I=1,KKK
0081
                        1-444=166
0082
                   112 WRITE(102,1040) GNAME, ENTRY(JJJ), UG(JJJ), VG(JJJ)
                        KKK=NG+1
DD 113 1=1,KKK
UAP([]=UG(KKK+1-[]
0083
0084
0085
                   113 VAP(1)=VG(KKK+1-1)
0086
0087
                        CALL HULLER (NG, UAP, VAP, NAPP, UAPP, VAPP, J, UZROS, VZROS, JAP, XSTART, XEN
                       ID, NOPOLY, URAPP, VRAPP I
IF(J.EQ. 0) GO TO 150
WRITE(102, 1180)
0088
0089
0090
                        IF(JAP.EQ.0) GO TO 120
0091
                        GO TO 130
                   115 KKK=NDO+1
0092
                        00 116 I=1,KKK
UD3([]=UD0([)
0093
0094
                  116 VD3([]=VD0([)
0095
                        ND 3= ND 0
0096
                        GO TO 100
```

```
0098
                    120 KKK=JAP+1
0099
                         WRITE(ID2,1085) (I,UZROS(I),VZROS([),J1,1=KKK,J)
                   GO TO 140
130 00 135 I=1,JAP
135 WRITE(102,1190) I,UZROS(I),VZROS(I),JI,URAPP(I,2),VRAPP(I,2)
IF(JAP,LT.J) GO TO 120
0100
0101
0102
0103
0104
                    140 IF(J.EQ.NG) GO TO 155
0105
                    150 WRITE(102,1095)
0106
                         IF(J.EQ.0) GD TO 170
                    155 DO 160 I=1,J
UROOT(KO+I)=UZROS(I)
0107
0108
0109
                         VRODT(KD+1)=VZROS(I)
0110
                    160 MULTIKO+[]=J1
0111
                         K={J*J1}+K
0112
                         KÐ=KD+J
0113
                         IF(K.GE.NP) GO TO 1
0114
                    170 J1=J1+1
0115
                         IF(ND1.LE.1) GO TO 200
0116
                         DG 180 [=1,ND]
                         UDO([]=UD1([)
VDO([]=VD1([)
0117
0118
                         UDDO(1)=UDD1(1)
0120
                    180 VDD0(1)=VDD1(1)
0121
                         U00 ( NDI +1 ) =U01 ( NDI +1 )
0122
                         V00(ND1+1)=VD1(ND1+1)
0123
                         NDO=ND1
                         NODO = NDO 1
0124
0125
                         KKK=NDZ+1
0126
                         00 190 1=1.KKK
0127
                         UD1(1)=UD2(1)
0128
                    190 VD1(I)=VD2(I)
                         ND1≈ND2
GD TO 20
0159
0130
0131
                   200 IFIND1: EQ. 01 GO TO 1
0132
0133
                         DENOM=U01(2)*U01(2)*VD1(2)*VD1(2)
                         URODT(KD)={-UD1(1)*UD1(2)-VD1(1)*VD1(2))/DENOM VROOT(KD)=(-VD1(1)*UD1(2)+UD1(1)*VD1(2))/DENOM
0134
0135
                         MULT(KD)=J1
0136
0137
                         WRITE(102,3000) (ASTER, [=1,33]
0138
                         WRITE(102,1035) JL
0139
                         KKK=ND1+1
0140
0141
0142
                        NNN=KKK+1
                         1-NMM=LLL
0143
                   210 WRITE(102,1100) DINAME, ENTRY (JJJ), UDICJJJ), VDICJJJI
0144
                         WRITE(102.1180)
                         WRITE(102,1085) KD, UROOT(KD), VROOT(KD), J1.
0145
                         60 TO 1
0146
                  1020 FORMATIIHI, 10x, 48HREPEATED USE OF THE GREATEST COMMON DIVISOR AND 1, 44, 44, 58H METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIA
0147
                  2LS/11x,18HPOLYNOMIAL NUMBER ,12///)
1025 FORMAT(///1x,25HNO ROOTS OF MULTIPLICITY ,12//)
1035 FORMAT(///1x,87HTHE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE R
100TS OF P1X) WHICH HAVE MULTIPLICITY ,12//)
1085 FORMAT(2X,5HROOT(,12,4H) = ,023.16.3H + ,023.16.2H 1.8X,12,9X,25HN
0148
0149
0150
                       10 INITIAL APPROXIMATIONS
                  1095 FORMATI///IX,51HNDT ALL ROOTS OF THE ABOVE POLYNOMIAL, G. HERE FOUN
0151
```

```
10//1
0152
                                  1000 FORMAT(3(12,1x),9x,13,1x,3(06,0,1x),20x,2(07,0,1x),11)
0153
                                  1010 FORMAT(2030.0)
0154
                                   1015 FORMAT(2030.0)
                                  1030 FORMAT(1x,22HTHE DEGREE OF P(x) IS ,12,22H THE COEFFICIENTS ARE//
0155
                                  1040 FORMAT(2X,A2,A2,4H) = ,D23.16,3H + ,D23.16.2H 1)
1100 FORMAT(2X,A3,A2,4H) = ,D23.16.3H + ,D23.16.2H I)
1180 FORMAT(///IX,13HROOTS OF P(X),52X,14HMULTIPLICITIES.17X,21HINITIAL
0156
0157
0158
                                1180 FORMAT(///IX,13HRODTS OF PIX),52X,14HMULTIPLICITIES.17X,21HINITIAL
1 APPROX(MATION//)
1190 FORMAT(2X,5HRODTI,[2,4H] = ,D23.16,3H + ,D23.16,2H 1,8X,12,8X,D23.
116,3H + ,D23.16,2H 1)
2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN. ,12)
2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,13)
2020 FORMAT(1X,29HRADIUS TO START SEARCH.,11X,D9.2)
2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,D9.2)
2050 FORMAT(1X,21HRADIUS TO END SEARCH.,11X,D9.2)
2060 FORMAT(//IX)
2070 FORMAT(//IX)
2070 FORMAT(//IX)
2070 FORMAT(1X,34HTEST FOR ZERO IN SUBROUTINE GCD. ,09.2)
2080 FORMAT(/////IX,33,32A4)
END
0159
0160
0161
0162
0163
0164
0166
0167
0168
                                               END
0169
```

```
0001
                              SUBROUTINE PRODUM.UF, VF.N. UG, VG. HN, UH, VHI
                    000000
                           * GIVEN POLYNOMIALS R(x) AND S(x). THIS SUBROUTINE COMPUTES THE * COEFFICIENTS OF THE PRODUCT POLYNOMIAL T(x) = R(x).S(x).
                              OOUBLE PRECISION UH, VH, UF, VF, UG, VG
DIMENSION UH(51), VH(51), UF(26), VF(26), UG(26), VG(26)
 0002
 0003
0004
0005
0006
                              MN=M+N
KKK=MN+1
                              DO 100 I=1.KKK
K=I
UH([]=0.0
0007
8000
0009
0010
0011
0012
                              VH(11=0.0
IF(1.LE.M+1) GO TO 10
LIMIT=M+1
                         GO TO 20
10 LIMIT≈1
0013
                         20 DD 50 J=L<sub>1</sub>LIMIT

IF(K.GT.N+L) GD TO 50

IF(J+K.EQ.1+L) GD TO 40

GO TO 50
0014
0015
0016
0017
0018
                         40 UH([]=UH([]+(UF(J)*UG(K)-VF(J)*VG(K))
                              VH(1)=VH(1)+(VF(J)*UG(K)+UF(J)*VG(K))
0019
                        50 K=K-1
100 CONTINUE
0020
0021
0022
0023
                              RETURN
                              END
```

```
0001
                          SUBROUTINE QUAD(N,UA,VA,J,UROOT, VROOT)
                 C
                         SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR. SOLUTION OF THE QUADRATIC IS DONE USING THE QUADRATIC FORMULA.
                      *
                 C
0002
                         DOUBLE PRECISION EPS1.EPS2.EPSLON, URGOT, VROOT, UA, VA, UDISC, VDISC, UD
                        1.VD.DDD,UTEMP,VTEMP, 868
DOUBLE PRECISION EPS4
DIMENSION UROOT(25),VROOT(25),UA(26),VA(26)
0003
0004
0005
                         COMMON EPSI, EPS2, EPSLON, EPS4, TO2, MAX
0006
                         IF(N.GT.1) GO TO 10
0007
                         J≖J+l
8000
                         BB8=UA(2)+UA(2)+VA(2)+VA(2)
0009
                         URDOT(J) =- (UA(1) +UA(2)+VA(1)+VA(2)) /BBB
0010
                         VROQT(J) =- (VA(1) *UA(2)-UA(1) *VA(2)]/8BB
001 L
                         GO TO 100
                     10 UDISC=[UA(2)+UA(2)-VA(2)+VA(2))-(4.0+(UA(3)+UA(1)-VA(3)+VA(1)))
0012
                         VD1SC=(2.0*UA(2)*VA(2))~(4.0*(UA(3)*VA(1)*VA(3)*UA(1)))
0013
0014
                         UD=2.0*UA(3)
0015
                         VD=2.0*VA(3)
0016
                         DDD=DSQRT(UDISC*UDISC+VDISC*VDISC)
                         IF(DOD-LT.EPSLON) GO TO 20
CALL COMSQT(UDISC, VOISC, UTEMP, VTEMP)
0017
0018
                         BBB=UD+UD+VD+VD

UROGT(J+1)=({-UA(2}+UTEMP}*UD+(-VA(2)+VTEMP)*VD]/BBB

VROOT(J+1)=({-VA(2}+VTEMP)*UD-(-UA(2)+UTEMP)*VD]/BBB

UROGT(J+2)=({-UA(2}-UTEMP)*UD+(-VA(2)-VTEMP)*VD]/BBB
0019
0020
0021
0022
                         VROOT(J+2)=((-VA(2)-VTEMP)*UD-(-UA(2)-UTEMP)*VD)/888
0023
0024
                         S+L=L
                         GO TO 100
0025
                         J=J+1
0026
0027
                         888=UD*UD+VD*VD
0028
                         UROOT(J)=(-UA(2)*UD-VA(2)*VD)/888
                         VROOT (J) = (-VA(2)*UD+UA(2)*VD)/888
0029
                  WRITE(102,1000) UROOT(J), VROOT(J)
1000 FORMAT(///IX,11HQUAD FOUND ,D23.16.3H + ,D23.16.2H 1,22H TO BE A M
0030
0031
                       IULTIPLE ROOT//)
0032
                   100 RETURN
0033
                         END
```

```
SUBROUTINE GCD(N,UR, VR, M, US, VS, MI, USS, VSS)
0001
             0000000
                   GIVEN POLYNOMIALS PIX) AND DPIX) WHERE DEG. DPIX) IS LESS THAN DEG. P(X), SUBROUTINE GCD COMPUTES THE GREATEST COMMON DIVISOR OF P(X) AND
                    DP (X).
                  DOUBLE PRECISION USSSSS.VSSSSS
0002
                    DOUBLE PRECISION UR, VR, US, VS, USS, VSS, URR, VRR, UD, VD, UT, VT, EPSLON, EP
0003
                   152, EP53, EP54, BBB
                    DIMENSION UR126), VR126), US1261, VS1261, USS1261, VSS1261, URR1261, VRR1
0004
                   1261,UT(261,VT(26)
0005
                    COMMON EPSLON, EPS2, EPS3, EPS4, 102, MAX
0006
                    N1=N
0007
                    MI=M
                    KKK=N+1
0008
                    DO 20 I=1.KKK
0009
0010
                    URR([]=UR([)
0011
                20 VRR(II=VR(II
0012
                    KKK=M+1
                    DO 25 I=1,KKK
USS(1)=US(1)
0013
0014
                   VSS(1)=VS(1)
0015
                 30 BB8=USS(H1+11+USS(H1+11+VSS(H1+11+VSS(H1+1)
0016
                    UD=(URR(N1+1)*USS(M1+1)+VRR(N1+1)*VSS(M1+1))/BBB
VD=(USS(M1+1)*VRR(N1+1)-URR(N1+1)*VSS(M1+1))/BBB
0017
0018
                    KKK=N1+1-M1
0019
                    DO 40 I=KKK.NL
0020
                    UT(1)=URR(1)-(UD*USS(1-N1+M1)-VD*VSS(1-N1+M1))
VT(1)=VRR(1)-(UD*VSS(1-N1+M1)+VD*USS(1-N1+M1))
IF(M1.EQ.N1) GQ TQ 70
0021
0022
0023
0024
                    KKK=N1-M1
                    DO 60 I=1 KKK
0025
                    UT(II=URR(I)
0026
                60 VT([)=VRR([)
0027
                 70 DO 90 (=1.N1
0028
                    BBB=DSQRT(UT(NI+1-11+UT(N1+1-11+VT(N1+1-1)+VT(N1+1-1))
0029
                    IFIBBLGT.EPSLONE GO TO 100
0030
                   CONTINUE
0031
                    DO 95 1=1.ML
0032
                    888=USS(M1+1)*USS(M1+1)+VSS(M1+1)*VSS(M1+11
0033
                    0034
0035
                    USS(11=USSSS
0036
                    V55(11=V55555
                 95
0037
                    USS(M1+1)=1.0
0038
                    V$5(MI+1)=0.0
0039
0040
                    GO TO 200
0041
               100 K=N1-E
                    IF(K.EQ.0) GO TO 170
0042
                    IF(K.LT.MI) GO TO 140
0043
                    KKK=K+1
0044
0045
                    DQ 130 J=1.KKK
                    URR(J) #UT(J)
0046
               130 VRR(J)=VT(J)
0047
                    N1≃K
0048
                    60 TO 30
0049
```

0050	140	KKK=K+1
0051		00 150 J=1.KKK
0052		URRIJ)=USS(J)
0053		VRR(J)=VSS(J)
0054		ひららしりゃけたり
0055	150	(L)1V≈(L)22V
0056		KKK=K+2
0057		NNN=ML+1
0058		DD 160 J=KKK,NNN
0059		URR(J1=USS(J)
0060	160	VRR(J)#VSS(J)
0061		NL=M1
0062		H1=K
0063		GD TO 30
0064	170	USS(1)=1.0
0065		V\$5(1)=0.0
0066		M1=0
0067	200	RETURN
0068		END

```
0001
                         SUBROUTINE DIVIDE(N.UP.VP.M.UD.VO.K.UQ.VQ)
                00000
                        GIVEN TWO POLYNOMIALS F(X) AND G(X), SUBROUTINE DIVIDE COMPUTES THE QUOTIENT POLYNOMIAL H(X) = F(X)/G(X).
0002
                         DOUBLE PRECISION UP. VP. UD. VD. UQ. VQ. UTERM. VTERM. UDUMMY
                         DIMENSION UP (26), VP (26), UD (26), VD (26), UQ (26), VQ (26)
0003
0004
0005
                        K=N-M
UDUMMY=UD(M+1)*UO(M+1}+YO(M+1)*YO(M+1)
                        UQ(K+1)=(VP(N+1)*UD(M+1)*VP(N+1)*VD(M+1))/UDUMMY
VQ(K+1)=(VP(N+1)*UD(M+1)-UP(N+1)*VD(M+1))/UDUMMY
0006
0007
0008
                         IF(K.EQ.O) GO TO 100
0009
                         DO 50 I=1.K
0010
0011
                         J + i = L
0012
                         UTERM=UP(N-J)
0013
                         VTERM=VP(N-J)
0014
                         KK=K+1
0015
0016
                         L-M=NNN
                        DO 40 ME=NNN,M
IF(KK.GT.1) GD TO 10
0017
0018
                         GO TO 45
0019
                    10 1F(M1.GE.1) GO TO 20
                    GO TO 40
20 UTERM=UTERM-IUQ(KK)*UD(M1)-VQ(KK)*VO(M1))
VTERM=VTERM-(UQ(KK)*VO(M1)+VQ(KK)*UD(M1))
0020
0021
0022
0023
                    40 KK=KK-1
0024
                    45 UDUMMY=UD(M+1)+UD(M+1)+VD(M+1)*VD(M+1)
                    UQ(K+1-[]=(UTERM*UD(M+1)+VTERM*VD(M+1))/UDUMMY
50 VO(K+1-1)=(VTERM*UD(M+1)-UTERM*VD(M+1))/UDUMMY
0025
0026
                   100 RETURN
0027
                         END
0028
```

```
SUBROUTINE CONSQTIUX, VX, UY, VY)
0001
                      00000
                                 THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.
                                DOUBLE PRECISION UX, WX, WY, DUMMY, R, AAA, 888

R=DSQRT(UX+WX+VX)

AAA=DSQRT(DABS((R+WX)/2.0))

BBB=DSQRT(AABS((R-WX)/2.0))

IF(VX) 10,20,30

IV=AAA
0002
0004
0005
9000
                           10 UY=AAA
VY=-1.0*88B
GO TO 100
20 IF(UX) 40.50.60
0007
0008
0009
0010
                           30 UY=AAA
YY=BBB
GD TO 100
40 DUMMY=DABS(UX)
UY=0.0
0011
0012
0013
0014
0015
0016
                                 VY=DSQRT(DUMMY)
                           GO TO 100

50 UY=0.0

VY=0.0

GO TO 100

60 DUMMY=DABS(UX)
0018
0019
0020
0021
0022
                                 UY=DSQRT(DUMMY)
0023
                                 VY=0.0
0024
                          100 RETURN
0025
                                 END
```

```
0001
                      SUBROUTINE CALCIUXI, VXI, UX2, VX2, UX3, VX3, UPXI, VPXI, UPX2, VPX2, UPX3, V
                     LP X3, UX4, VX4, UQ4, VQ4, UH3, VH31
                    ***********************************
              ç
                      GIVEN THREE APPROXIMATIONS X(N-Z), X(N-L), AND X(N), SUBROUTINE CALC APPROXIMATES THE POLYNOMIAL BY A QUADRATIC AND SOLVES FOR THE ZERO OF THE QUADRATIC CLOSEST TO X(N). THIS ZERO IS THE NEW APPROXIMATION
              C
              C
                      XIN+L) TO THE ZERO OF THE POLYNOMIAL.
                                                    ***********
0002
                      DOUBLE PRECISION ARGI, ARG2
                     DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UX1,VX1,UX2,VX2,UX3,VX3,UPX1, 1VPX1,UH3,VH3,UH2,VH2,UQ3,VQ3,UD,VD,UB,VB,UC,VC,UDISC,VDISC,VDISC,VCC,VC
0003
                     2CC, UDENI, VDENI, UDEN2, VDEN2, UQ4, VQ4, UX4, VX4, EPSRT, EPSQ, EPS, UODD, VDD 3D, AAA, BBB, RAD, UAAA, VAAA, UBBB, VBBB
                      DOUBLE PRECISION THETA, ANGLE, UTEST, VYEST DOUBLE PRECISION EPS1
0004
0005
0006
                      COMMON EPS1, EPS, EPS0, EPSRT, 102, MAX
0007
                      UH3=UX3-UX2
ODOR
                      VH3=VX3-VX2
                      UH2=UX2-UX1
0009
DOLO
                      VH2=VX2-VX1
0011
                      BBB=UH2*UH2*VH2*VH2
0012
                      UQ3=(UH3*UH2+VH3+VH2)/888
0013
                      VQ3=(VH3*UH2~UH3*VH2)/BBB
0014
                      U0DD=1.0+UQ3
                      VDDD=VQ3
0015
0016
                      UD=(UPX3-(UDQ0*UPX2-VDDD*VPX2))+(UQ3*UPX1-VQ3*VPX1)
0017
                      VD=( VPX3-( VDDD*UPX2+UDDD*VPX2) I+( VQ3*UPX1+UQ3*VPX1)
0018
                      UAAA=2.0#UQ3
                      VAAA=2.0*VQ3
UAAA=UAAA+1.0
0019
0020
                      UBBB=UDDD+UDDO-VDDD+VDDD
0021
                      V888=V000*U000+U000*VDDD
0022
0023
                      UCCC=UQ3+UQ3-VQ3+VQ3
0024
                      VCCC=VQ3*UQ3+UQ3*VQ3
                      U8=({UAAA+UPX3-VAAA+VPX3}-LUB08+UPX2-V8B8+VPX2}}+{UCCC+UPX1-VCCC+V
0025
                     1PX11
0026
                      V8={(VAAA*UPX3+UAAA*VPX3)-(V888*UPX2+UBBB*VPX2)}+(VCCC*UPX]+UCCC*V
                     1PX1}
                      UC=UDDD+UPX3-VDDD+VPX3
0027
                      VC=VD0D*UPX3+U00D*VPX3
0028
                      UDISC=(UB+UB-VB+VB)-(4.0+(UD+UC-VD+VC))
0029
                      VD[SC=[2.0*(V8*UB]]-[4.0*(VD*UC+UD+VC]]
0030
                      AAA=DSQRT(UDISC+UDISC+VDISC+VDISC)
0031
0032
                      LETAMA.EQ.O.OT GO TO 5
0033
                      GO TO 7
                   5 THETA=0.0
0034
                      60 TO 9
0035
                     THETA=DATAN2(VOISC, UDISC)
0036
                     RAD=DSQRT(AAA)
0037
                      ANGLE=THETA/2.0
0038
                      UTEST=RAD+DCOS(ANGLE)
VTEST=RAD+DSIN(ANGLE)
0039
0040
                     UDEN1=UB+UTEST
0041
0042
                      VDEN1=V8+VTEST
0043
                      UDEN2=UB-UTEST
0044
                      VDENZ=VB-VTEST
```

```
ARG1=UDENI*UDEN1+VDEN1*VDEN1
ARG2=UDEN2*UDEN2*VDEN2*VDEN2
AAA=DSQRTIARG1)
BBB=DSQRT(ARG2)
IFIAAA.LT.BBB) GD TD 10
IF(AAA.EQ.Q.Q) GO TO 60
UAAA=-2.0*VC
UQ4=(UAAA*UDEN1*VAAA*VDEN1*/ARG1
VQ4=(VAAA*UDEN1-VAAA*VDEN1)/ARG1
GD TO 50
10 IF(BBB.EQ.Q.Q) GD TO 60
UAAA=-2.0*VC
UAA=-2.0*VC
VAAA=-2.0*VC
VAAA=-2.0*VC
VAAA=-2.0*VC
UQ4=(UAAA*UDEN2*VAAA*VDEN2)/ARG2
VQ4=(VAAA*UDEN2-UAAA*VDEN2)/ARG2
     0045
0046
0047
0048
     0049
          0050
       0051
     0052
0053
0054
     0055
0056
0057
0058
0059
     0060
     0061
     0062
     0063
                                                                                                                                                                                                                                            RETURN
60 UQ4=1.0
VQ4=0.0
GO TO 50
END
0064
0065
0066
  0067
     8800
```

```
SUBROUTINE MULLER (NP, UA, VA, NAPP, UAPP, VAPP, NROOT, UROOT, VROOT, TROOT,
0001
                         1XSTART,XEND,NOPOLY,URAPP,VRAPP)
                 00000
                       * MULLER'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A * POLYNOMIAL OF MAXIMUM DEGREE 25. THROUGH THREE GIVEN POINTS THE * POLYNOMIAL IS APPROXIMATED BY A QUADRATIC. THE ZERO OF THE QUADRATIC * CLOSEST TO THE OLD APPROXIMATION IS TAKEN AS THE NEW APPROXIMATION.
                  ¢
                          IN THIS MANNER A SEQUENCE IS OBTAINED CONVERGING TO A ZERO.
                         DOUBLE PRECISION UPX3, VPX3, UPX2, VPX2, UROOT, VROOT, UX1, VX1, UAPP, VAPP 1, UX2, VX2, UWURK, VWORK, UX3, VX3, UB, VB, UX4, VX4, UA, VA, UPX1, VPX1, URAPP, V
0002
                         ZRAPP, UPX4, YPX4, EPSRT, EPSB, EPS, CCC, EPSM, UH3, VH3, UQ4, VQ4, ABPX4, ABPX3
                         3,QQQ,XSTART,XEND
                         DIMENSION URODT(25), VROOT(25), MULT(25), UAPP(25,3), VAPP(25,3), UWORK
1(26), VWORK(26), UB(26), VB(26), UA(26), VA(26), URAPP(25,3), VRAPP(25,3)
LOGICAL CONV
0003
0004
                           COMMON EPSM. EPS. EPSO. EPSRT. 102, MAX
0005
                           DATA PNAME, DNAME/2HP(, 2HD(/
0006
                           EPSM=0.0000
0007
8000
                           EPSRT=0.999
0009
                           NRBOT=0
0010
                           !RODT=0
0011
                           IPATH=1
0012
                           NOMULT=0
                           NALTER=0
0013
0014
                           ITIME=0
0015
                           [APP#1
                           ITER=1
0016
0017
                           IF(NAPP.NE.O) GO TO 18
                           NAPP=NP
0018
0019
                           CALL GENAPP(UAPP, VAPP, NAPP, XSTART)
                      GO TO 27
18 DO 25 I=1.NAPP
UAPP([,1]=0.9*UAPP([,2]
0020
0021
0022
                           VAPP(1,1)=0.9*VAPP(1,2)
0023
0024
                           UAPP(1,3)=1.1*UAPP(1,2)
0025
                      25 VAPP(1,3)=1.1*VAPP(1,2)
                      27 KKK=NP+1
OG 30 [=1,KKK
UWORK(I)=UA(I)
0026
0027
0028
                          VWDRK(1)=VA(1)
0029
0030
                          NWORK=NP
                          UX1=UAPP(IAPP,1)
VX1=VAPP(IAPP,1)
UX2=UAPP(IAPP,2)
0031
0032
0033
                           VX2=VAPP([APP,2]
0034
                           UX3=UAPP([APP.3]
0035
0036
                           VX3=VAPP([APP.3]
                          CALL HORNER(NWORK, UWORK, UWORK, UX1, VX1, UB, VB, UP X1, VPX1)
CALL HORNER(NWORK, UWORK, VWORK, UX2, VX2, UB, VB, UPX2, VPX2)
CALL HORNER(NWORK, UWORK, VWORK, UX3, VX3, UB, VB, UPX3, VPX3)
0037
0038
0039
                      SO CALL CALCIUXI, VXI, UX2, VX2, UX3, VX3, UPX1, VPX1, UPX2, VPX2, UPX3, VPX3, UX
0040
                         14, VX4, UQ4, VQ4, UH3, VH3)
                          CALL HORNER (NWORK, UWORK, VWORK, UX4, VX4, UB, VB, UPX4, VPX4)
0041
0042
                          0043
```

```
IF(ABPX3.EQ.0.0) GO TO 70
0044
0045
                       QQQ=ABPX4/ABPX3
                        (F(QQQ.LE.10.) GO TO 70
0046
                       UQ4=0.5+UQ4
0047
0048
                       UX4=UX3+[UH3+UQ4-VH3+VQ4]
0049
0050
                       VX4=VX3+[VH3*UQ4+UH3*VQ4)
                   GO TO 60

70 CALL TESTIUX3, VX3, UX4, VX4, CONV)

IFICONV) GO TO 120

IFILITER.LT.MAX) GO TO 110
0051
0052
0053
0054
                      CALL ALTER(UAPP(IAPP,1), VAPP(IAPP,1), UAPP(IAPP,2), VAPP(IAPP,2), UAP
1P(IAPP,3), VAPP(IAPP,3), NALTER, ITIME)
0055
0056
                        IF(NALTER-GI-5) GO TO 75
                   ITER=1
GO TO 40
75 [F(IAPP.LT.NAPP) GO TO 100
1F(XEND.EQ.0.0) GO TO 77
0057
005B
0059
0060
                       IF(XSTART.GT.XEND) GO TO 77
0061
0062
                       NAPPENP
                       CALL GENAPP(UAPP, VAPP, NAPP, XSTART)
0063
                       IAPP=0
0064
                       GO TO 100
0065
0066
                    77 WRITE(102,1090)
0067
                       KKK=NWORK+1
                       WRITE(102,1035) (DNAME, J. UWORK(J), VWORK(J), J=1,KKK)
0068
                   80 IF(NROOT.EQ.O) GO TO 90
IF(IPATH.EQ.1) GO TO 82
0069
0070
0071
                       IPATH=2
                       CALL SETTER (UA, VA, NP, UROOT, VROOT, NROOT, URAPP, VRAPP, [ROOT, MULT]
0072
0073
                       RETURN
                   82 IF(NROOT.EQ.01GO TO 90
IF(1ROOT.EQ.0) GO TO 85
WRITE(102,1080)
0074
0075
0076
                       DO 55 1=1, IROUT
0077
                    55 WRITE(102,1085) [,UROOT(1),VROOT(1),URAPP(1,2),VRAPP(1,2)
0078
                        IFIIRDOT.LT.NROOT) GO TO 85
0079
                   GO TO 87
85 KKK=1800T+1
0080
0081
                        WRITE(102,1086) (1,UROOT(1),VROOT(1),I=KKK,NROOT)
0082
                    87 IF(IPATH.EQ.L1 GO TO 81
0083
0084
                       RETURN
                    90 WRITE(102:1070) NOPOLY
0085
                       RETURN
0086
                  100 IAPP=IAPP+1
0087
                        ITER=1
0088
                       NALTER=0
0089
0090
                        GO TO 40
                  120 NROOT=NROOT+1
0091
                        IRDOT=NROOT
0092
                        MULT (NROOT) = 1
0093
                        NOMULT = NOMULT+1
0094
0095
                       UROOT (NROOT) =UX4
0096
                       VROOT (NROOT) = VX4
                       URAPP(NROUT, 1) = UAPP(IAPP, 1)
VRAPP(NROUT, 1) = VAPP(IAPP, 1)
URAPP(NROUT, 2) = UAPP(IAPP, 2)
0097
0098
0099
                        VRAPP(NROOT, 2) = VAPP( IAPP, 2)
0100
```

```
0101
                        URAPPINROOT, 31 = UAPPIIAPP, 3)
0102
                        VRAPP(NROOT, 3)=VAPP( [APP, 3)
0103
                   125 IF(NONULT.LT.NP) GO TO 130
0104
                        GO TO 80
                  130 CALL HORNER (NWORK, UWORK, VWORK, UX4, VX4, UB, VB, UPX4, VPX4)
NWORK=NWORK-1
0105
0106
                        KKK=NWORK+1
                        DO 140 I=1.KKK
0108
                  UWDRK(1)=UB11)

140 YWORK(1)=VB(1)

CALL HORNER (NWORK, UWORK, UX4, VX4, UB, VB, UPX4, VPX4)

CCC=DSQRT(UPX4*UPX4+VPX4*VPX4)

IF(CCC, LT, EPSM) GO TO 150
0109
0110
0111
0112
0113
                        IF(NWORK.GT.2) GO TO 75
                       IROOT=NRODT

KKK=NWORK+1

DD 145 I=1,KKK

UB(I)=UWORK(KKK+1-I)
0115
0116
0117
0118
0119
                  145 V8(I)=VWDRK(KKK+1-I)
0120
                        CALL QUAD(NWORK, UB, VB, NROOT, UROOT, VROOT)
0121
                        G0 TO 80
                  150 MULT(NROOT)=MULT(NROOT)+1
0122
                        NOMULT=NOMULT+1
0123
                        GO TO 125
0124
0125
                   110 UX1=UX2
0126
                        VX1=VX2
0127
                        UX2=UX3
0128
                        VX2≃VX3
0129
                        UX3=UX4
                        VX3=VX4
0130
0131
                        UPX1=UPX2
0132
                        VPX1=VPX2
0133
                        UPX2=UPX3
                        VPX2=VPX3
0134
                        UPX3=UPX4
0135
0136
                        VPX3=VPX4
0137
                        ITER=ITER+I
                 GO TO 50

1090 FORMAT(///,1X,65HCOEFF[CIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO LZEROS WERE FOUND//)

1080 FORMAT(///IX,13HROOTS OF G(X),83X,21HINITIAL APPROXIMATION//)
0138
0139
01.40
                 1070 FORMAT(//,43H NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER ,12)
1086 FORMAT(2x,5HROOT(,12,4H) = ,023.16,3H + ,023.16,2H I.19x,23HSOLVED
0141
0142
                 0143
0144
0145
0146
                        END
```

.3.2

```
1000
                  SUBROUTINE GENAPPIAPPR, APPI, NAPP, XSTART)
            00000
                  SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE DEGREE OF THE ORIGINAL POLYNOMIAL.
               0002
0003
0004
0005
0006
0007
0008
0009
0010
0011
0012
0013
0014
0015
0016
0017
               20 APPI(I:3)=1.1*APPI(I:2)
0018
                  RETURN
0019
                  END
```

```
.0001
                          "SUBROUTENE ALTEREXER.XLL.X2R.X2I.X3R.X3I.NALTER, LTIME!
                  00000
                           SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT.
0002
                           DOUBLE PRECISION XIR, XII, X2R, X2I, X3R, X3I, EPS1, EPS2, EPS3, R, BETA
                           DOUBLE PRECISION EPSM
COMMON EPSM.EPS1,EPS2,EPS3,102,MAX
IF(ITIME.NE.O) GO TO 5
0003
0004
0005
0006
                           ITIME=1
                        WRITE(102,1010) MAX
5 IF(NALTER.EQ.0) GO TO 10
WRITE(102,1000) X1R,X11,X2R,X21,X3R,X31
0007
0008
0009
                      GO TO 20
10 R=DSQRT[x2R+x2R+x21+x21)
0010
0011
                           RETA=0ATAN2(X21,X2R)
WRITE({02,1020| X1R,X11,X2R,X21,X3R,X31
0012
0013
0014
                       20 NALTER=NALTER+1
0015
                            IF(NALTER.GT.5) RETURN
                           GO TO (30,40,30,40,30),NALTER
0017
                       30 X2R=-XZR
0018
                           X21=-X21
0019
                           GO TO 50
0020
                       40 BETA=BETA+1.0471976
0021
                           X2R=R*DCOS(BETA)
                      X21=R+DSIN(BETA)
50 XIR=0.9*XZR
0022
0023
0024
                           X11=0.9*X21
0025
                            X3R=1.1*XZR
0026
                           X3I=1.1*X21
0027
                           RETURN
                   1000 FORMAT(1x,5Hx1 = ,023.16,3H + ,023.16,2H I,10x,22HALTERED APPROXIM
1ATIONS/1x,5Hx2 = ,023.16,3H + ,023.16,2H I/1x,5Hx3 = ,023.16,3H +
2,023.16,2H I/)
0028
                   1020 FORMAT(1H0,5HX1 = ,D23.16,3H + ,D23.16,2H [,10X,22HINITIAL APPROXI

1MAT(0NS/1X,5HX2 = ,D23.16,3H + ,D23.16,2H [/1X,5HX3 = ,D23.16,3H +
0029
                   2 ,D23.16,2H I/)
1010 FORMAT(///1x,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
1TER ,13,12H ITERATIONS.//)
0030
0031
```

 $= c_{A} C_{p} \tilde{t}^{A}$ 

```
0001
                                                          SUBROUTINE BETTER (UA.VA.NP. URGOT, VROOT, HROOT, URAPP, VRAPP, LROOT, HUL
                                       C
                                      CCC
                                                          SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND
                                                         BY USING THEM AS INITIAL APPROXIMATIONS WITH MULLER'S METHOD APPLIED TO
THE FULL, UNDEFLATED POLYNOMIAL.
                                      C.
                                                                               DOUBLE PRECISION UROOT. VROOT, UA. VA. UBAPP. VBAPP. UX1. VX1. UX2. VX2. UX3
0002
                                                      1. VX3, UPX1, VPX1, UPX2, VPX2, UPX3, VPX3, UB, VB, UROOTS, VROOTS, EPSRT, UX4. V
2X4, URAPP, VRAPP, EPSO, EPS, UQ4, VQ4, UH3, VH3
 0003
                                                         DOUBLE PRECISION EPSM
 0004
                                                      OTHERSION UROOT(25), VROOT(25), UA(26), VA(26), UBAPP(25,3), VBAPP(25,3), UBAPP(25,3), VBAPP(25,3), VBAPP(25,
                                                          LOGICAL CONV.
 0005
                                                         COMMON EPSH, EPS, EPSO, EPSRT, 102, MAX
0006
 0007
                                                          IF(NROOT.LE.1) RETURN
 0006
                                                          L=0
                                                         OO 10 [=1,NROOT
UBAPPII,1)=UROOT([]=EPSRT
VBAPP[],1)=VROOT([]=EPSRT
0009
 0010
0011
 0012
                                                          UBAPP(1,2)=UROOT(1)
 0013
                                                          VBAPP(I.2)=VROOT(I)
                                                        UBAPP(1.3)=URGGT(1)*(2.0-EPSRT)
VBAPP(1.3)=VRGGT(1)*(2.0-EPSRT)
DG 100 J=1,NRGGT
0014
0015
0016
0017
                                                          UX1=UBAPP(j.1)
 00 LB
                                                          VXI=VBAPP(J.1)
                                                         UX2=UBAPP(J.2)
VX2=VBAPP(J.2)
 0019
0020
                                                          UX3=U8APP(J,3)
0021
                                                          VX3=VBAPP(J.3)
0022
                                                          ITER=1
0023
                                               CALL HORNER (NP.UA, VA.UX1.VX1.UB, VB.UPX1.VPX1)
CALL HORNER (NP.UA, VA.UX2.VX2.UB, VB.UPX2.VPX2)
20 CALL HORNER (NP.UA, VA.UX3.VX3.UB, VB.UPX3.VPX3)
CALL CALC (UX1.VX1.UX2.VX2.UX3.VX3.UPX1.VPX1.UPX2.VPX2.UPX3.VPX3.UX
14.VX4.U04.VQ4.UH3.VH3)
 0024
0025
0026
0027
                                                30 CALL TEST(UX3,VX3,UX4,VX4,CONV)
                                                        IF(CONV) GO TO 50
IF(ITER,LT.MAX) GO TO 40
WRITE(102,1000) J.UROOT(J),VROOT(J),MAX
WRITE(102,1010) UX4,VX4
IF(J.LT.IROOT) GO TO 33
IF(J.EQ.IROOT) GO TO 35
0029
0030
0031
0032
0033
0034
                                                       GO TO 100
KKK=[RDOT-1
0035
0036
                                                        DO 34 K=J.KKK
URAPP(K,1)=URAPP(K+1,1)
VRAPP(K,1)=VRAPP(K+1,1)
0037
0038
0039
0040
                                                         URAPP(K.2)=URAPP(K+1.2)
                                                         VRAPP(K, 2)=VRAPP(K+1, 2)
0041
                                                        URAPP(K,3)=URAPP(K+1,3)
VRAPP(K,3)=VRAPP(K+1,3)
0042
0043
                                                        IROOT=IROOT-1
0044
```

0045

GO TO 100

```
0046
0047
0048
                                40 UX1≈UX2
                                     VX1=VX2
UX2=UX3
VX2=VX3
0049
0050
                                      UX3=UX4
0051
                                      ¥X3=¥X4
0052
                                     UPX1=UPX2
0053
0054
0055
                                     VPX1≃VPX2
                                     UPX2=UPX3
VPX2=VPX3
1TER=ITER+1
0056
0057
                                     GO TO 20
0058
0059
0060
                               50 L=L+1
                             UROOTS(L1=UX4
VROOTS(L1=VX4
100 CONTINUE
0061
                            100 (CONTINUE

1F(L.EO.O) GO TO 120

DO 110 1=1,t

UROOT([]=UROOTS([])

110 VROOT([]=VROOTS([])

NROOT=L
0062
0063
0064
0066
                            RETURN
120 NRODT=0
0067
8400
0069
                                     RETURN
                          1000 FORMAT(///42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,12,4H) = .
1023.16,3H + .023.16,2H I/24H DID NOT CONVERGE AFTER .13,11H ITERAT
210NS)
1010 FORMAT(30H THE PRESENT APPROXIMATION IS .023.16,3H + .023.16,2H I/
0070
0071
                                   1/}
END
0072
```

```
0001
                            SUBROUTINE TESTILUX3, VX3, UX4, VX4, CONV)
                            SUBROUTINE TEST CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-
                            IMATIONS BY TESTING THE EXPRESSION
                           ABSOLUTE VALUE OF (X(N+1)-XIN))/ABSOLUTE VALUE OF X(N+1). WHEN IT IS AS SMALL AS DESIRED. CONVERGENCE IS OBTAINED.
                           DOUBLE PRECISION UX3, VX3, UX4, VX4, EPSRT, EPSO, EPS, AAA, UDUMMY, VOUMMY,
 0002
                          IDENOM
0003
                           DOUBLE PRECISION EPSM
LOGICAL CONV
COMMON EPSM, EPS, EPSO, EPSRT, 102, MAX
0004
0005
0006
0007
                           UDUMMY=UX4-UX3
                           VDUMMY = VX4 - VX3
                           AAA=DSSRTIUDUMMY*UDUMMY*VDUMMY*VDUMMY)
DENOM=DSGRT(UX4*UX4*VX4*VX4)
IF(DENUM.LT.EPSD) GO TO 20
IF(AAA/DENOM.LT.EPS) GO TO 10
0008
0009
0010
0011
0012
0013
                        5 CONVELFALSE.
                      GO TO 100

10 CONV=.TRUE.

GO TO 100

20 IF(AAA.LT.EPSO) GO TO 10
0014
0015
0.016
0017
                           GO TO 5
0018
                     100 RETURN
0019
                           END
0001
                           SUBROUTINE HORNER(NA, UA, VA, UX, VX, UB, VB, UPX, VPX)
```

C HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A POINT O. \*
SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE \* c FACTOR (X-D). DOUBLE PRECISION UX. VX. UPX. VPX. UB. VB. UA. VA DIMENSION UA(26), VA(26), UB(26), VB(26) 0002 0003 UB(1)=UA(1) VB(1)=VA(1) NUM=NA+1 0004 0005 0006 DO 10 1=2,NUM UB(I)=UA(I)+(UB(I-I)+UX-VB(I-I)+VX) 0007 0008 0009 10 V8([]=VA([]+(V8([-])#UX+UB([-])#VX) UPX=UB(NUM) 0010 0011 VPX=VBINUM1 0012 RETURN 0013 END